TAMPEREEN TEKNILLINEN YLIOPISTO
TAMPERE UNIVERSITY OF TECHNOLOGY

YURY ZHELEZOVSKIY

DSC-KALORIMETRIN SUUNNITTELU FAASIMUUTOSMATERIAALEILLE

Diplomityö

Tarkastaja: professori Jukka Vanhala
Tarkastaja ja aihe hyväksytty
Tieto- ja sähkötekniikan tiedekunta-neuvoston kokouksessa 9. marras-kuuta 2016

TIIVISTELMÄ

YURY ZHELEZOVSKIY: DSC-kalorimetrin suunnittelu faasimuutosmateriaaleille
Tampereen teknillinen yliopisto
Diplomityö, 52 sivua, 32 liitesivua
Tammikuu 2017
Tietotekniikan diplomi-insinöörin tutkinto-ohjelma
Pääaine: Sulautetut järjestelmät
Tarkastaja: professori Jukka Vanhala

Avainsanat: faasimuutos, faasimuutosmateriaali, kalorimetri, DSC-kalorimetri, lämmönsiirto, energia, lämpö

Työssä toteutettiin DSC-kalorimetri, jonka avulla pystytään tarkkailemaan faasimuutosmateriaaleja ja niiden ominaisuuksia. Työn tavoitteena oli suunnitella ja toteuttaa DSC-kalorimetri ja tutustua PCM-materiaaleihin.

Työn teoriaosuudessa käsitellään lämmönsiirron muotoja ja faasimuutoksia pinnallisesti. Lisäksi tutustutaan faasimuutosmateriaaleihin, niiden tyyppeihin ja sovelluskohteisiin. Työssä selitetään tavallisen kalorimetrin tehtävä ja toiminta. Lisäksi tarkastellaan tarkemmin, miten toimii DSC-kalorimetri, miten se eroaa tavallisesta kalorimetristä, ja mistä se koostuu.

Ongelmana oli se, että tällä hetkellä ei ole olemassa standardoitua tapaa mitata PCM-materiaalien sisältäviä tekstiilejä kalorimetrillä. On olemassa eri kalorimetrejä ja faasimuutosmateriaaleja, mutta niiden yhteinen standardisoitu toiminta vielä puuttuu.

Kalorimetrinen järjestelmä koostuu kolmesta moduulista ja niiden välisistä väylistä. PC-yksikkö on kokonaan toteutettu ohjelmallisesti Visual Studio-ympäristössä C# kielellä. Se piirtää reaaliajassa lämpötila-aika kuvaajaa, ja tarvittaessa se pystyy kalibroimaan lämpöantureita. PC-yksikkö kommunikoi keskusyksikön kanssa käyttäen UART-sarjaliikennettä. Keskusyksikkö asettaa mittausyksikössä olevat parametrit halutuiksi, ja kommunikoi laitteen kanssa SPI-väylän kautta. Mittausyksikkö suorittaa tarkat lämpömittaukset ja lämmittää tutkittavaa PCM-materiaalia. Kaikki mittaukset tapahtuvat uunissa, jossa on käytetty polyuretaania eristemateriaalina.

Mittauksissa tarkistettiin laskettu alumiinilevyn ja teräslämpövastuksen lämpökapasiteetti mittaamalla. Uunissa tapahtuva lämpövuoto riippuu lineaarisesti ympäristön ja uunin lämpötilaerosta. Mittausten ja laskelmien perusteella todettiin, että PCM-materiaali on sitonut enemmän energiaa (noin 80,72 J) faasimuutoksen aikana kuin ei-PCM. Lisäksi PCM-materiaali sitoo 38,44 Joulea grammaa kohti.

ABSTRACT

YURY ZHELEZOVSKIY: Design of DSC-calorimeter for phase change materials
Tampere University of Technology
Master of Science Thesis, 52 pages, 32 Appendix pages
January 2017
Master's Degree Programme in Information Technology
Major: Embedded systems
Examiner: Professor Jukka Vanhala

Keywords: phase change, phase change material, calorimetry, DSC-calorimeter, heat transfer, energy, heat

In this thesis, DSC-calorimeter was carried out, which can help observe phase change materials and their characteristics. The purpose of this work was to design and realize DSC-calorimeter, and become familiar with PCM-materials.

The fundamental modes of heat transfer and phase changes are discussed superficially in the theory part of this thesis. In addition, phase change materials, their types and their targets of application are become acquainted with. In this thesis a basic calorimeter's function and action are explained. In addition, how DSC-calorimeter works, and what difference between it and basic calorimeter, are analyzed more closely.

Currently there are no standard modes to measure PCM-materials using calorimeter, and it was a problem. There are some calorimeters and phase change materials, but their common standard operation is missing.

The calorimeter system consists of three modules and channels between them. PC-Unit (PC-yksikkö) are realized totally programmatically in Visual Studio-environment using C# program language. It is drawing in real time a temperature-time chart, and it is able to calibrate temperature sensors on demand. PC-Unit communicates with Centre-Unit (Keskusyksikkö) by using UART serial communication. Centre-Unit sets up integrated circuits, which are in Measurement-Unit (Mittausyksikkö). The communication between Centre-Unit and Measurement-Unit is SPI. Measurement-Unit performs exact heat measurements, and it is able to heat researching PCM-materials. All measurements are running in a furnace, which is done from insulating material called polyurethane.

The measurements have revised calculated aluminum plate and steel heating element by measuring the heat of the heating system. Furnace heat loss depends linearly on the environment and the furnace temperature difference. It was found that the PCM material is bound more energy (about 80,72 J) during the phase transition than non-PCM. In addition, the PCM material binds 38.44 Joules per gram.

ALKUSANAT

Muutimme vaimoni kanssa Suomeen vuonna 2011 paluumuuttajina. Ennen muuttoamme en osannut suomea ollenkaan. Kolmen vuoden Suomessa asumisen jälkeen hain Tampereen teknilliseen yliopistoon. Yllättäen pääsin opiskelemaan suoraan maisteri-vaiheeseen oman kotimaani diplomityön ja suomen kielen testin perusteella.

Pääaineeni, Sulautetut järjestelmät, koostuu sekä tietotekniikan että sähkötekniikan osuudesta. Täytyy sanoa, että elektroniikka ei ollut minulle tuttu alue, vaan päinvastoin. Sen takia ensimmäinen opiskelu vuosi tuotti minulle tuskia.

Vasta toisen vuoden aikana aloin ymmärtää kuinka kiinnostavaa alaa minä opiskelen. Ja loppujen lopuksi kahdessa vuodessa olin suorittanut kaikki opinnot.

Diplomityöpaikan etsimisessä oli paljon haasteita. Sen takia haluaisin kiittää suuresti tarkastajaani professori Jukka Vanhalaa, joka tarjosi minulle mahdollisuuden tehdä diplomityöni Tampereen Teknillisen Yliopiston Elektroniikan ja tietoliikennetekniikan laitoksella syksyn 2016 aikana. Sain paljon tukea, neuvoja ja ohjauksia häneltä, joista kiitän häntä erikseen vielä kerran. Intohimoni aiheeseen kasvoi työn edetessä rajusti ylöspäin.

Suuret kiitokset kuuluvat tietenkin vaimolleni ja meidän lapsillemme, jotka ovat tukeneet minua jaksamaan kaikki opiskeluun liittyvät paineet.

Tampereella, 1.12.2016

Yury Zhelezovskiy

SISÄLLYS

1. JOHDANTO 1
2. LÄMMÖNSIIRTO 2
2.1 Lämmönsiirron muodot 2
2.1.1 Johtuminen 2
2.1.2 Konvektio 3
2.1.3 Lämpösäteily 4
3. FAASIMUUTOS 6
3.1 Kiinteä ja neste 7
3.2 Neste ja kaasu 7
3.3 Kaasu ja kiinteä 7
3.4 Faasidiagrammi 8
4. FAASIMUUTOSMATERIAALI 10
4.1 Energia faasimuutoksen aikana 10
4.2 Faasimuutosmateriaalien määrittely 12
4.3 Faasimuutosmateriaalin tyypit 13
4.4 Faasimuutosmateriaalin sovelluskohteita 13
5. KALORIMETRIN SUUNNITTELU 14
5.1 Kalorimetrin määrittely 14
5.2 DSC kalorimetri 15
5.3 Ongelma 16
5.3.1 Kalorimetrin vaatimukset 17
5.4 Järjestelmän rakenteellinen kuvaus 18
5.4.1 PC-yksikkö 19
5.4.2 Keskusyksikkö 19
5.4.3 Mittausyksikkö 21
6. KALORIMETRIN TOTEUTUS 22
6.1 Väylät 22
6.1.1 PC-yksikön ja Keskusyksikön välillä tiedonsiirto 22
6.1.2 Keskusyksikön ja Mittausyksikön välillä tiedonsiirto 23
6.2 PC-yksikön toteutus 25
6.2.1 Ohjelmisto 26
6.3 Keskusyksikön toteutus 26
6.3.1 Ohjelmisto 27
6.4 Mittausyksikön toteutus 29
6.4.1 Lämpöanturin valinta 30
6.4.2 Mikropiirin tarve ja valinta 32
6.4.3 Mikropiirin ja anturin kytkentä 33
6.4.4 Lämpöluku 34
6.4.5 Antureiden kalibrointi 35
6.4.6 Levyjen lämmitykseen tarvittava teho 37

6.4.7 PWM 40
6.5 Muut komponentit 41
6.6 Fyysinen toteutus 41
6.6.1 Piirilevy 42
6.6.2 Lämpöanturit 43
6.6.3 Eristeuuni 43
7. MITTAUSTULOKSET 44
7.1 Mittaus 1 44
7.1.1 Mittaustulokset 44
7.1.2 Tulosten pohdinta 45
7.1.3 Päätelmä 45
7.2 Mittaus 2 45
7.2.1 Mittaustulokset 46
7.2.2 Tulosten pohdinta 47
7.2.3 Päätelmä 47
7.3 Mittaus 3 48
7.3.1 Mittaustulokset 48
7.3.2 Tulosten pohdinta 49
7.3.3 Päätelmä 51
8. YHTEENVETO 52

LIITE A: MITTAUSYKSIKÖN TOTEUTUS

LIITE B: MITTAUSYKSIKÖN KYTKENTÄKAAVIO

LIITE C: MITTAUSYKSIKÖN LAYOUT TOP

LIITE D: MITTAUSYKSIKÖN LAYOUT BOTTOM

LIITE E: KESKUSYKSIKÖN OHJELMA

LIITE F: PC-YKSIKÖN OHJELMA

LYHENTEET JA MERKINNÄT

DSC	engl. Differential Scanning Calorimeter, differentiaalinen pyyhkäisykalorimetri
IDE	engl. Integrated development environment
MISO	engl. master in slave out
MOSI	engl. master out slave in
PCM	engl. Phase change material, faasimuutosmateriaali
PT100	engl. Platinum resistor
PWM	engl. Pulse with modulation, pulssinleveysmodulaatio
RTD	engl. Resistance temperature detector, vastuslämpötila-anturi
SCK	engl. serial clock
SI-järjestelmä	ransk. Système international d'unités, kansainvälinen mittayksikkö-järjestelmä
SPI	engl. Serial peripheral interface, sarjareuna rajapinta
TTY	Tampereen teknillinen yliopisto
USART	engl. Universal synchronous/asynchronous receiver/transmitter

λ	lämmönjohtavuus
Q	lämpöenergia
W	työ
m	massa
T	lämpötila
t	aika
c	ominaislämpökapasiteetti
C	lämpökapasiteetti
Lf	sulamislämpö, jähmettymislämpö
Lv	höyrystymislämpö, tiivistymislämpö
V	jännite
I	virta
R	resistanssi
P	teho
ρ	tiheys
k	lämmönläpäisykerroin

1. JOHDANTO

Energialla on maailmanlaajuinen tarve, sitä käytetään paljon ja sen tarve kasvaa koko ajan. Tästä johtuen uusiutuva energia ja sen käyttö, merkitsee paljon ja koskee kaikkia. Parhaillaan kehitetään ja yritetään löytää uusia tapoja, miten energiaa voitaisiin tuottaa helpommin ja enemmän, sekä miten sitä pystyttäisiin säästämään. Se on tätä päivää. Energiaa voidaan varastoida, ja sillä on myös iso rooli globaalissa maailmassa, ja siihen on useita syitä.

Työn tarkoituksena on tutustua faasimuutosmateriaaleihin, jotka voivat varastoida energiaa ja vapauttaa sitä. Näillä materiaaleilla on paljon sovelluskohteita ja tulevaisuus näyttää lupaavalta. Tutustuminen näihin materiaaleihin on vain osa tätä diplomityötä. Koko työn ytimenä on faasimuutosmateriaaleja varten toteutetun kalorimetrin suunnittelu ja valmistus. Kalorimetrin avulla on mahdollista tutkia niitä materiaaleja, joilla on merkitystä eri aloilla niiden käyttöä ajatellen. Järjestelmä koostuu eri moduuleista, joista kerrotaan tarkemmin myöhemmin.

Tässä diplomityössä tutustutaan ensin teoreettisiin asioihin ja vasta sen jälkeen suunnitteluun. Luvussa 2 kerrotaan lämmönsiirrosta ja sen muodoista yleisellä tasolla. Seuraavaksi luvussa 3 selitetään materiaalien faasimuutoksia. Faasimuutosmateriaaleista puhutaan luvussa 4. Luvussa kerrotaan, mitkä ne ovat, missä niitä käytetään jne. Kalorimetrin järjestelmän rakenteellinen kuvaus on esitetty luvussa 5. Moduuleiden kommunikaatioon käytetty väylä on kuvattu luvussa 6. Samassa luvussa kuvaillaan moduuleiden fyysinen ja ohjelmallinen toteutus. Mittaustulokset on koostettu lukuun 7. Lopuksi luvussa 8 on yhteenveto koko diplomityöstä.

2. LÄMMÖNSIIRTO

Termodynamiikka on tiede, jonka keskipisteessä on energia ja sen muutosten säännönmukaisuus. Termodynamiikka kuvaa miten lämpö siirtyy systeemin sisällä tai systeemien välillä. [5] Näiden systeemien eri lämpötilat ja keskinäiset vuorovaikutukset aiheuttavat lämmönsiirron. Termodynaamiseksi systeemiksi kutsutaan materiaalisten kappaleiden kokonaisuutta, jossa kappaleet kommunikoivat sekä keskenään että ympäristön kanssa.

Termodynamiikassa termi lämpö voi tarkoittaa energian muotoa. Tällöin vaatimuksena on energian kulkeutuminen, vasta silloin lämpöä voidaan pitää energian muotona.

Lämmönsiirto tai lämmön siirtyminen on kappaleen sisäisen energian muutoksen yksi tapa. Tällöin kappaleen sisäinen energia siirtyy toisen kappaleen sisäiseen energiaan ilman mekaanista työtä. [34]

Lämmönsiirto tapahtuu systeemien välillä tietyissä olosuhteissa ja se ei tapahdu satunnaisesti, vaan se noudattaa aina fysikaalisia sääntöjä. Suunnitellessaan projekteja insinöörin on kiinnitettävä huomiota, miten ja mistä lämmönsiirto tapahtuu, koska joissain tapauksissa lämmönsiirto voi olla merkittävä tekijä.

Termodynamiikan mukaan lämmönsiirrolla on eri muotoja. Jokainen muoto noudattaa omaa ehtoa tai lakia ja tapahtuu vasta silloin kun ehdot toteutuvat.

Lämmönsiirto yleensä jaetaan perinteisesti kolmeen muotoon, jotka ovat johtuminen, konvektio ja lämpösäteily. [32]

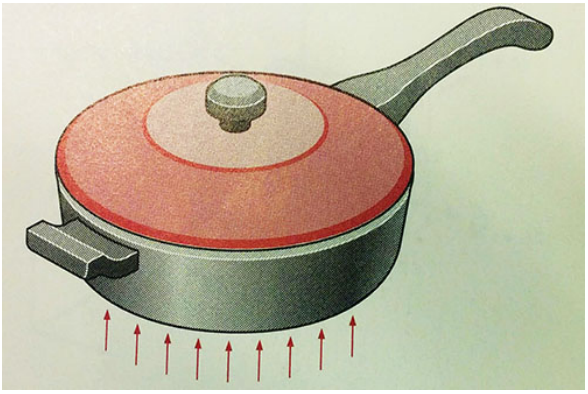
Reaaliolosuhteissa lämpö voi siirtyä samaan aikaan käyttäen kaikki kolmea muotoa.

2.1 Lämmönsiirron muodot

Tässä kappaleessa tutustutaan kaikkiin muotoihin tarkemmin.

2.1.1 Johtuminen

Johtuminen (engl. *Conduction*) on eri osien lämpötilaerojen aiheuttamaa lämmönsiirtymistä. Johtuminen voi tapahtua sekä kiinteissä ja nestemäisissä aineissa että kaasuissa. Käytännössä lämpö siirtyy lämpimämmästä osasta kylmempään. Kuvassa 1 johtuminen noudattaa samaa periaatetta, eli lämpö siirtyy lämpimämmästä osasta (liedestä) kylmempään (pannuun) [8]. Samaan aikaan lämpö siirtyy pannun pinnalta pannulla olevaan ruokaan.



Kuva 1. *Johtuminen [8]*

Johtuminen aiheutuu mikrohiukkasten (atomien, molekyylien) lämmönsiirrosta ja vuorovaikutuksesta. Tällöin kappaleen lämpötila pyrkii tasaantumaan.

Lämmönjohtavuus (engl. *thermal conductivity*) (λ) kuvaa kuinka hyvin tai huonosti aine tai materiaali johtaa lämpöä. Eri materiaalilla on oma lämmönjohtavuuden arvo. Lämmönjohtavuuden yksikkö on *W/(K·m)*.

Metalleilla on korkeimmat lämmönjohtavuus arvot. Nesteillä arvot ovat matalammat kuin metalleilla ja matalimmat arvot ovat kaasuilla. Esimerkiksi alumiinin, veden ja ilman lämmönjohtavuuden arvot huonelämpötilassa (25°C) ovat seuraavat:

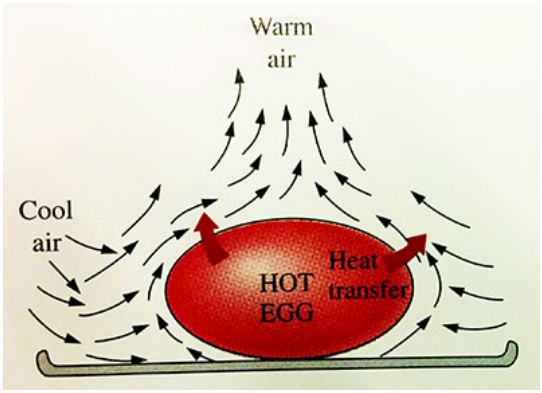
$\lambda_{Al} = 205\ \frac{W}{K \cdot m}, \lambda_{H_2O} = 0{,}58 \frac{W}{K \cdot m}$ ja $\lambda_{ilma} = 0{,}024 \frac{W}{K \cdot m}$ [30]

Sen takia, jos halutaan kantaa astiaa, jossa on kuumaa nestettä, ei kannata käyttää välissä märkää liinaa. Kuivan liinan kuitujen välissä on vain ilmaa ja märän liinan kuitujen välissä on vettä, jonka lämmönjohtavuus on suurempi kuin ilman. Sen takia märkä liina johtaa paremmin lämpöä ja polttaa käsiä.

2.1.2 Konvektio

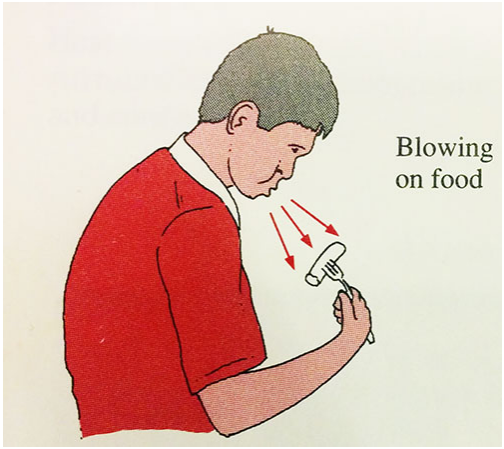
Konvektio (engl. *Convection*) on lämmönsiirron muoto, jossa lämpö siirtyy nesteen tai kaasun mukana. Konvektiolla on kaksi tyyppiä: **luonnollinen konvektio** (engl. *natural convection*) ja **pakotettu konvektio** (engl. *forced convection*). [8]

Luonnollisessa konvektiossa lämpö siirtyy kappaleesta ilman ulkoista apua. Kuvassa 2 lämpö siirtyy kuumasta kananmunasta lämpimän höyryn mukana ilmaan. [8]



Kuva 2. *Luonnollinen konvektio [8]*

Pakotetussa konvektiossa lämpö siirtyy kappaleesta jonkin ulkoisen voiman avulla. Kuvassa 3 lämpö siirtyy kuumasta nakista ilmaan ihmisen puhaltaman viileän ilmavirran ansiosta. [8]



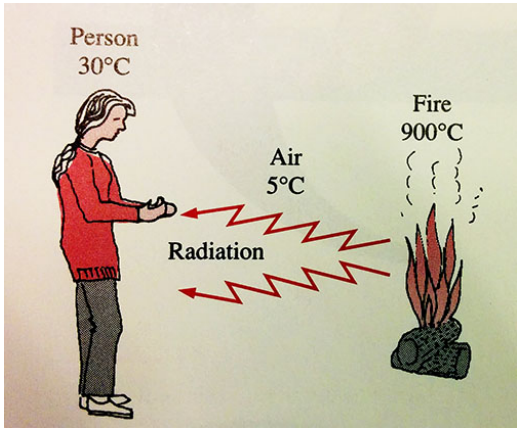
Kuva 3. *Pakotettu konvektio [8]*

Konvektiota ei voi olla kiinteissä aineissa ja konvektion suuruus riippuu nesteen tai kaasun lämpötilaerosta.

Konvektio-ilmiön takia kodeissa lämpöpatterit on sijoitettu ikkunoiden alapuolelle ja ikkunat yläpuolelle. Kevyempi lämmin ilma nousee ylös ja painavampi kylmä ilma laskee alas. Näin ollen konvektion ansiosta saadaan aikaan yksinkertainen painovoimainen ilmastointi.

2.1.3 Lämpösäteily

Lämpösäteily (engl. *Radiation*) on sähkömagneettista säteilyä, joka syntyy aineen omasta energiasta. Sähkömagneettisen säteilyn aallonpituudet vaihtelevat. Energia siirtyy sähkömagneettisten aaltojen mukana, joita ovat esimerkiksi auringon säteet. Silloin niitä kutsutaan lämpösäteilyksi. Kuvassa 4 lämpösäteily lähtee tulesta ja osuu ihmiseen [8].



Kuva 4. *Lämpösäteily [8]*

Lämpösäteily on mahdollista missä tahansa aineessa ja jopa tyhjiössä. Materiaalit pystyvät sekä säteilemään, että sitomaan säteilyä. Tummat materiaalit sitovat säteilyä paremmin kuin vaaleat. Musta kangas ei juurikaan heijasta valoa, mistä johtuen kaikki siihen osuvat säteilyt sitoutuvat siihen ja muuttuvat lämmöksi. Tämän ominaisuuden ansiosta tummissa vaatteissa on lämpimämpää kuin vaaleissa.

Lämpösäteilyn ansiosta tavalliset mikroaaltouunit lämmittävät ruokaa. Mitä enemmän ruuassa on vettä, sitä nopeammin se lämpenee, koska vesi pystyy sitomaan hyvin sähkömagneettisia aaltoja mikroaaltouunin taajuudella.

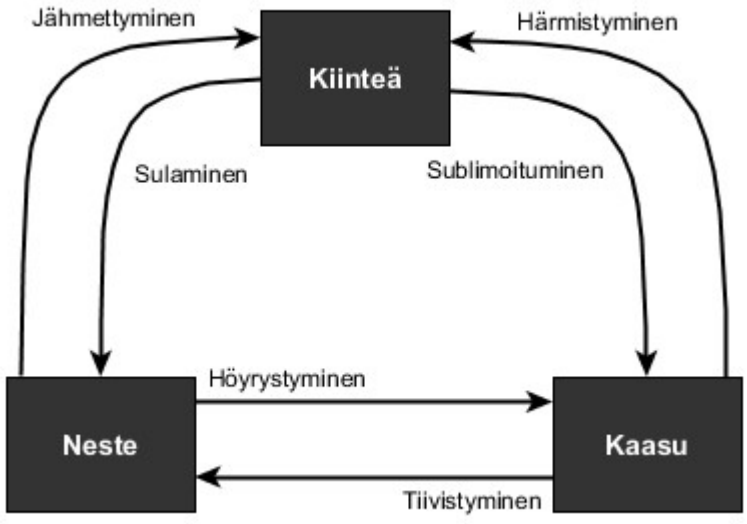
3. FAASIMUUTOS

Tietyssä lämpötilassa ja tietyssä paineessa eri aineilla on eri olomuodot. Huoneenlämmössä ja huoneilman paineessa kupari on kiinteä, vesi on neste ja typpi on kaasu. Olosuhteiden muuttuessa aineiden olomuodot voivat myös muuttua. [9]

Suuri lämmönsiirto voi aiheuttaa faasimuutoksen, kun materiaali, tai pikemminkin sen sisällä oleva aine muuttaa olomuotoaan. Faasimuutoksen aikana materiaalin lämpötila ei muutu ja kaikki energia kuluu vain materiaalin faasimuutokseen, eli materiaalin olomuoto vaihtuu.

Siirrettyä lämpömäärää (tunnus Q) kuvaa yksikkö nimeltään kalori, cal (*calorie*). Se on kuitenkin vanha energian yksikkö, joka ei virallisesti kuulu enää SI-järjestelmään. Kalorin sijaan käytetään toista yksikköä nimeltään Joule, J, koska lämpö on periaatteessa energiaa. Joule kuuluu SI-järjestelmään. 1 cal = 4,18J. [18] Nykyään kalori-yksikköä käytetään vain, kun puhutaan elintarvikkeiden energiasta.

Seuraavaksi tutustutaan lyhyesti yleisiin faasimuutoksiin. Yleensä faasimuutoksilla tarkoitetaan aineen olomuotojen muutoksia. Aineen yleiset olomuodot ovat neste, kaasu ja kiinteä. Aineen olomuoto voi muuttua miksi tahansa uudeksi olomuodoksi. Kuvassa 5 on esitetty olomuotojen väliset muutokset: härmistyminen, sublimoituminen, tiivistyminen, höyrystyminen, sulaminen ja jähmettyminen.



Kuva 5. *Aineen olomuotojen muutokset*

3.1 Kiinteä ja neste

- **Sulaminen** (engl. *melting*).
  Sulaminen on aineen muutos kiinteästä olomuodosta nesteeksi. Esimerkki: jää muuttuu vedeksi ympäristön lämpötilan kasvaessa.

- **Jähmettyminen** (engl. *freezing*).
  Jähmettyminen on sulamisen vastakohta, eli aineen muutos nestemäisestä olomuodosta kiinteäksi. Esimerkki: vesi muuttuu jääksi ympäristön lämpötilan laskiessa.

3.2 Neste ja kaasu

- **Höyrystyminen** (engl. *vaporization*).
  Höyrystyminen on aineen muutos nestemäisestä olomuodosta kaasuksi. Esimerkki: vesi muuttuu höyryksi, kun sitä lämmitetään tarpeeksi.

- **Tiivistyminen** (engl. *condensation*).
  Tiivistyminen on höyrystymisen vastakohta, eli aineen muutos kaasusta nesteeksi. Esimerkki: sumuisessa säässä höyry usein muodostaa nestemäisiä pisaroita.

3.3 Kaasu ja kiinteä

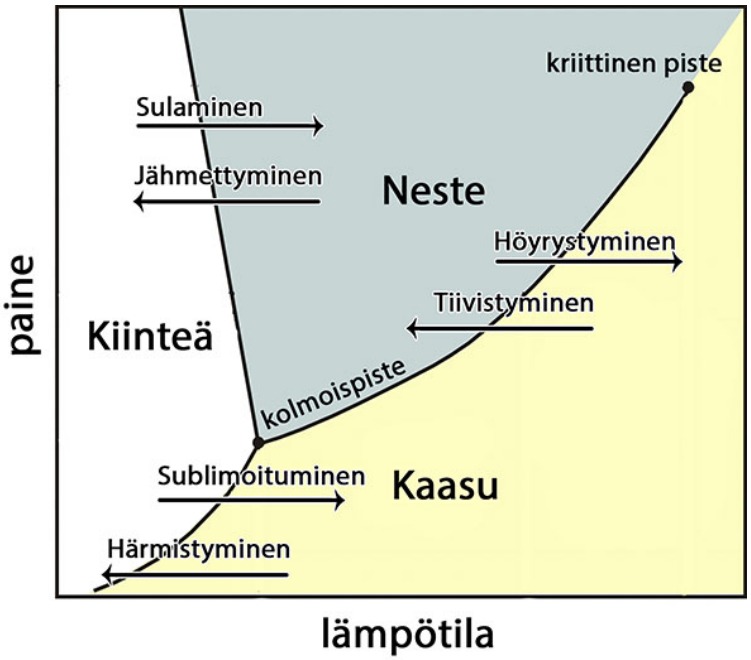
- **Sublimoituminen** (engl. *sublimation*).
  Sublimoituminen on aineen muutos kiinteästä olomuodosta suoraan kaasuksi ilman nestemäistä välitilaa.

- **Härmistyminen** (engl. *deposition*).
  Härmistyminen on sublimoitumisen vastakohta, eli aineen muutos kaasusta suoraan kiinteäksi. Esimerkki: lähes sama tilanne kuin tiivistymisen esimerkissä, mutta pakkasessa höyry muuttuu heti kiinteään olomuotoon eli kuuraksi.

Yllä mainitut faasimuutokset kuuluvat siihen joukkoon, jossa lämpötilan ja/tai paineen muuttuessa materiaalin ominaistilavuus, sisäenergia tai jokin muu suure muuttuu.

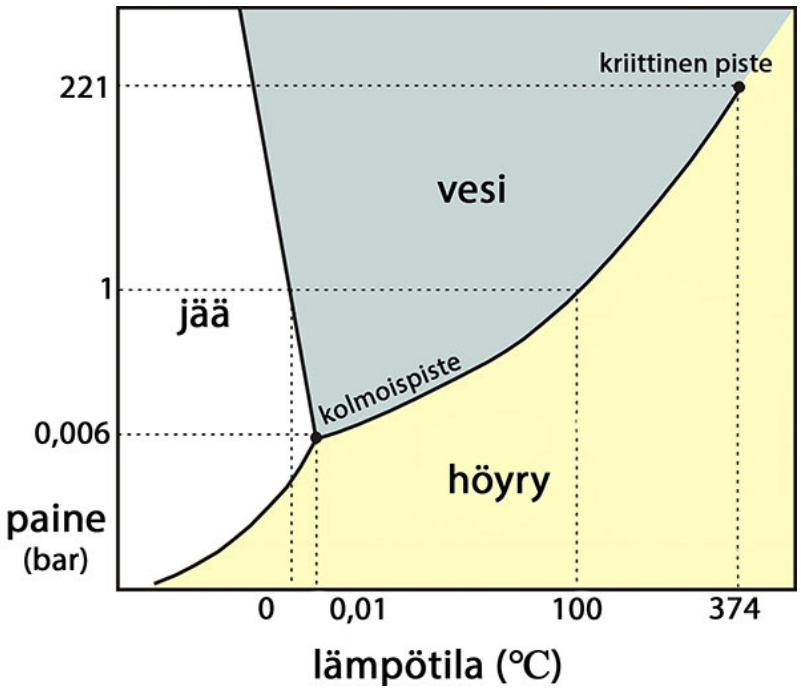
3.4 Faasidiagrammi

Olomuotojen muutoksia auttaa ymmärtämään faasidiagrammi, kuva 6.



Kuva 6. *Faasidiagrammi*

Kuvasta näkyy, että faasimuutos tapahtuu, jos paine (engl. *pressure*) ja/tai lämpötila (engl. *temperature*) muuttuu. Jokaisella aineella on oma kriittinen piste (engl. *critical point*) ja kolmoispiste (engl. *triple point*).



Kuva 7. *Veden faasidiagrammi*

Jos lämpötila ylittää kriittisen pisteen lämpötilan, kaasua on mahdotonta tiivistää millään paineella. Eli vaikka paine kasvaisi kuinka paljon, se ei riitä muuttamaan kaasua nesteeksi. Höyryä voidaan muuttaa nesteeksi, jos sen lämpötila on kriittisen pisteen lämpötilan alapuolella. Esimerkiksi kuvassa 7 näkyy, että veden kriittinen piste on 657,15K (374 °C) ja 22,1MPa (221 bar). [31]

Kolmoispisteessä kaikki faasit ovat tasapainossa. Siinä tilassa ainetta voidaan muuttaa nesteeksi, kaasuksi tai kiinteäksi lämpötilan ja/tai paineen pienen muutoksen avulla. Esimerkiksi veden kolmoispiste on 273,16K (0,01 °C) ja 611Pa (0,00611 bar). [11]

4. FAASIMUUTOSMATERIAALI

4.1 Energia faasimuutoksen aikana

Työssä tutustuttiin siihen, miten lämpö siirtyy kappaleista tai kappaleiden välillä käyttäen eri lämmönsiirron muotoja. Näin saatiin käsitys, miten lämmönsiirto vaikuttaa faasimuutoksiin, joista kuuteen perehdyttiin.

Puhuttaessa faasimuutosten tyypeistä energian käyttäytyminen jätettiin käsittelemättä. Aikaisemmassa luvussa se tehtiin siitä syystä, että olisi helpompaa ymmärtää faasimuutoksia ylipäätään. Lähestyttäessä faasimuutosmateriaalia täytyy ymmärtää mitä tapahtuu energiassa faasimuutoksen aikana.

Jos halutaan tietää, kuinka paljon energiaa tarvitaan materiaalin lämmittämiseen tietyllä lämpövälillä, voidaan käyttää kaavaa 4.1. [33]

$$Q = m \cdot c \cdot \Delta T \tag{4.1}$$

jossa m on materiaalin **massa** (kg), c on **ominaislämpökapasiteetti** (engl. *specific heat capacity)* (J/kg·K) ja ΔT on lämpötilan **muutos** (K).

Kaavaa voidaan käyttää, jos kappale tai materiaali on tehty vain yhdestä aineesta.

Ominaislämpökapasiteetti kerrottuna materiaalin massalla on lämpökapasiteetti. Tämä on esitetty kaavassa 4.2.

$$C = m \cdot c \tag{4.2}$$

Lämpökapasiteetti (engl. *heat capacity*) on kappaleen kyky varastoida energiaa tai lämpöä. Lämpökapasiteetin merkki on **C**-kirjain, sen yksikkö on J/K.

Tällöin energian kaava saa lyhyemmän muodon, joka on esitetty kaavassa 4.3

$$Q = C \cdot \Delta T \tag{4.3}$$

Näin ollen, jos pitää lämmittää yksi litraa vettä, kun alku- ja loppulämpötila on 10℃ ja 30 °C, voidaan laskea, kuinka paljon energiaa tarvitaan veden lämmittämiseen.
Ratkaisu siihen on seuraavan lainen. Yksi litra vettä painaa noin yhden kilogramman ja veden ominaislämpökapasiteetti on 4182 J/kg℃ [29], tällöin kaavan mukaan $Q = m \cdot c \cdot \Delta T = 1kg \cdot 4182 \frac{J}{kg℃} \cdot 20℃ = 83{,}64\ kJ$.

Yllä esitetty kaava ei päde kokonaan, jos kyseessä on aine, jonka faasi muuttuu.

Sulamisen aikana tarvitaan energia, jota kutsutaan **sulamislämmöksi** (engl. *heat of fusion*) [10]. Tarkemmin sanottuna sulamislämpö kuvaa kuinka paljon energiaa aineen massayksikköä tai ainemäärää kohti tarvitaan aineen sulamiseen. Sen yksikkö on J/kg tai sitten J/mol, ja tunnus on $\mathbf{L_f}$.

Jähmettymisen aikana energiaa ei tarvita, vaan sitä vapautuu faasimuutoksen aikana. Tätä energiaa massayksikköä kohti kutsutaan **jähmettymislämmöksi**. Periaatteessa jähmettymislämpö on sama kuin sulamislämpö, mutta siinä energiaa vapautuu, kun sulamisessa sitä sitoutuu [19]. Sen merkki on sama, eli $\mathbf{L_f}$.

Näin ollen faasimuutoksen aikana energia vapautuu (jähmettyminen) tai sitoutuu (sulaminen). Näiden kahden energiat voidaan laskea kaavalla 4.4. [19].

$$Q = \pm m \cdot L_f \tag{4.4}$$

Näin ollen, jos pitää ensin sulattaa yksi kilogramma jäätä ja lämmittää sitä, kun alku- ja loppulämpötila on 0℃ ja 20℃, voidaan laskea, kuinka paljon energia tarvitaan jään sulamiseen ja veden lämmittämiseen kokonaisuudessaan.
Ratkaisu siihen on seuraavan lainen. Aikaisemman esimerkin mukaan veden lämmittämiseen tarvitaan $83{,}64\ kJ$ energiaa. Mutta nyt pitää ottaa huomioon jään sulaminen. Jään sulamislämpö ($L_{f.}$) on 334kJ/kg [28]. Ja kun kyseessä on yksi kilogramma, silloin $Q = 1kg \cdot 334 \frac{kJ}{kg} = 334\ J$. Kokonaisenergia on sitten 83,64kJ + 334kJ = 417,64 kJ.

Höyrystymislämpö (engl. *heat of vaporisation*) on energia, joka tarvitaan muuttamaan neste kaasuksi [19], ja **tiivistymislämpö** on energia, joka vapautuu, kun kaasu muuttuu nesteeksi. Sen merkki on sama, eli L_v. Höyrystymislämpö lasketaan kaavan 4.5 mukaan.

$$Q = \pm m \cdot L_v \tag{4.5}$$

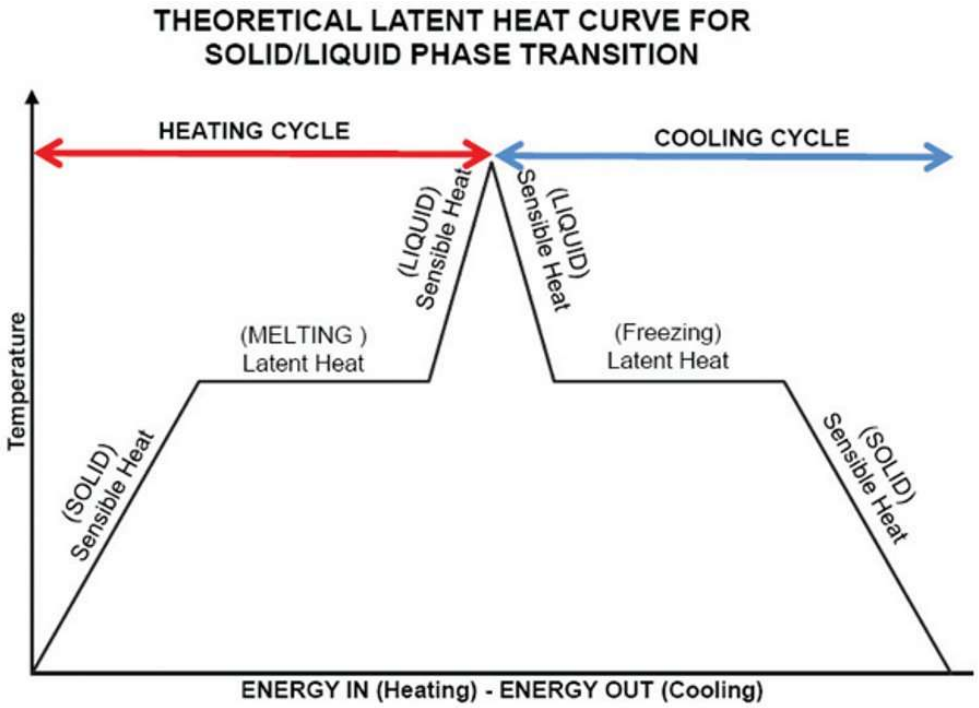
4.2 Faasimuutosmateriaalien määrittely

Faasimuutosmateriaali (engl. *phase change material, PCM*) on materiaali, jota käytetään lämmön varastoimisessa [24]. Termiä käytetään, kun puhutaan materiaaleista, jotka käyttävät hyväksi omaa faasimuutostaan energian varastoinnissa. [23]

Esimerkiksi vettä/jäätä voidaan pitää faasimuutosmateriaalina. Jään muuttumista vedeksi, eli sulamista, voidaan käyttää hyväksi lämmön varastoinnissa. Sulamisen aikana energiaa ei vapaudu, vaan sitä energiaa tarvitaan sulamiseen. Näin ollen jää sitoo lämpöä, koska jään sulamispiste on 0°C. [23] Tämän takia jään avulla jäähdytetään erilaisia juomia.

Esimerkiksi maanviljelijät käyttävät joskus veden muutosta jääksi, eli jähmettymistä hyväkseen. Tiedetään, että jähmettymisessä lämpöä vapautuu, jonka takia maanviljelijät ruiskuttavat lämpömuutokselle herkkiä kasveja vedellä ennen kylmää yötä (lämpötila laskee alle 0 °C). Vesi muuttuu jääksi ulkolämpötilan ollessa 0°C ja se vapauttaa 334J/g [28]. Tällöin lämpötila pyrkii pysymään 0 °C:ssa, eikä laske sen alle.

Faasimuutosmateriaaleja voidaan pitää latenttina lämpövarastoyksikkönä. [23]



Kuva 8. *Lämpötila-energian muutos [23]*

Kuvassa 8 näkyy, miten PCM-materiaalin lämpötila käyttäytyy ideaalisessa tapauksessa. Ensin lämpötila kasvaa lineaarisesti, kun syötetään energiaa, eli materiaali lämpenee. Sitten, kun lämpötila on saavuttanut aineen sulamispisteen (engl. *melting poin*), sulaminen alkaa. Kuvasta nähdään selvästi, että lämpötila ei enää kasva, vaan kaikki energia menee

faasimuutokseen, jota käsiteltiin luvussa 3. Tällöin materiaali varastoi energiaa. Kun materiaali on saanut tarpeeksi energiaa faasimuutokseen, sen lämpötila alkaa kasvaa.

Vastaprosessi alkaa, kun ympäristön lämpötila alkaa laskea alaspäin. Sitten materiaalin lämpötila saavuttaa jähmettymispisteen (engl. *freezing point*), ja materiaali alkaa vapauttaa varastoitua energiaa.

Kuvasta näkyy, että materiaali ns. latautuu ja purkautuu, kun faasi muuttuu kiinteästä nesteeksi ja päinvastoin. Tällöin materiaalin lämpötila pysyy vakiona.

4.3 Faasimuutosmateriaalin tyypit

Tässä diplomityössä käydään vain pinnallisesti läpi faasimuutosmateriaalien tyyppejä, koska se ei ole tämän työn ydintä. Tärkeimmät faasimuutosmateriaalien ryhmät ovat nykyään suolahydraatit ja parafiinit. [12]

Suolahydraatit ovat eniten tutkittuja lämpövarastointimateriaaleja. Niiden hinta on yleensä alhainen, koska niiden sisältö koostuu $M \cdot nH_2O$, jossa M on epäorgaaninen suolayhdiste. [23]

Parafiinit sopivat hyvin sovelluksiin, joissa on tietty lämpöväli. Niillä on hyvä lämmönvarastointikapasiteetti. [23]

4.4 Faasimuutosmateriaalin sovelluskohteita

Faasimuutosmateriaalien sovelluskohteita on useita, mutta tässä diplomityössä keskitytään tärkeimpiin ja suosituimpiin sovelluksiin, kuten rakennuksiin, kuljettamiseen ja tekstiileihin. [12]

- **Rakennukset**.
  PCM-materiaalien suosio rakennusalalla kasvaa. Niitä voidaan käyttää seinien tai kattojen eristämiseen. Faasimuutosmateriaalit mahdollistavat tasaisen huoneenlämmön.

- **Kuljettaminen**.
  Tämä sovelluskohde koskee eniten lämpöherkkiä materiaaleja. Joskus kuljetuksessa olevien tavaroiden (lääkkeet) on oltava tasapainoisessa ympäristössä.

- **Tekstiilit**.
  PCM-materiaalit ovat hyödyllisiä vaatteissa, jotka arvioivat ihmisen lämmön tarvetta.

5. KALORIMETRIN SUUNNITTELU

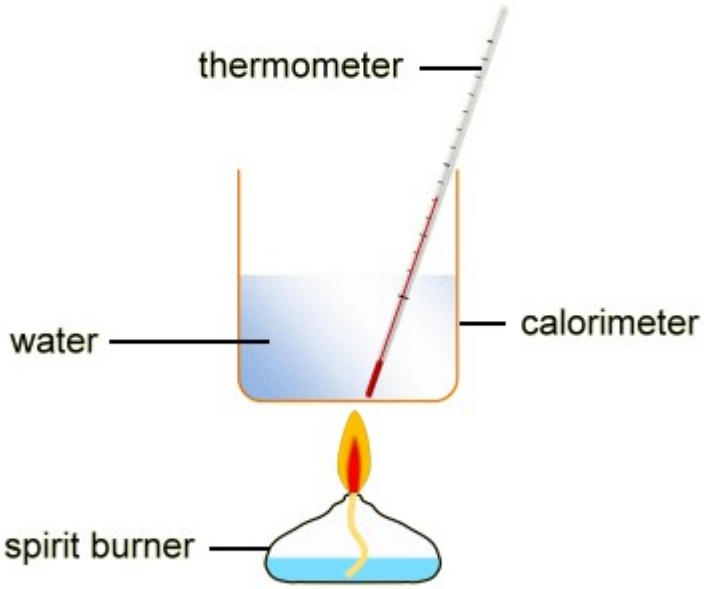
5.1 Kalorimetrin määrittely

Kalorimetri (lat. *calor* – lämpö, *metro* - mitata) on laite, jolla voidaan mitata materiaalin lämpömäärää, jota materiaali sitoo tai vapauttaa. [21]. Mittaus auttaa tarkkailemaan materiaalin lämpömuutoksia, faasimuutoksia, kemiallisia reaktioita jne. [22]

Kalorimetrin avulla on saatu aikaan pohja uusia teknologioita varten, kuten höyrykattila (engl. *steam boiler*), turbiini jne. [21]

Kalorimetri auttaa ymmärtämään, mitä materiaalissa tapahtuu, kun sitä lämmitetään. [25] Kalorimetreissä eniten tutkittu aine on vesi, koska se on yksinkertainen materiaali, jonka kanssa on helppo toimia, ja sitä on helppo saada. [21]

Kalorimetrin tärkein tehtävä on lämmön mittaus. Lämmön mittaus tarkoittaa lämpötilan muutoksen tarkkailemista. [14]



Kuva 9. *Yksinkertainen kalorimetri [7]*

Kuvassa 9 on esitetty yksinkertainen kalorimetri ja sen toiminta. Tässä tutkimuskohde on vesi, jonka alkulämpötila tiedetään. Vettä lämmitetään spriikeittimen avulla ja myöhemmin veden loppulämpötila otetaan talteen. Spriikeitin punnitaan ennen koetta ja kokeen jälkeen, jotta saadaan selville käytetyn polttoaineen määrä. [7]

Kalorimetrin avulla voidaan laskea materiaalin lämpökapasiteetti, tai käytetty energia käyttäen luvussa 4 esitettyä kaavaa 4.1 $Q = m \cdot c \cdot \Delta T$.

Kalorimetrejä on olemassa eri tyyppisiä, esimerkiksi: [21]

- **Reaktio kalorimetri** (engl. *reaction calorimeter*).
  Reaktio kalorimetri on kalorimetri, joka mittaa lämpöenergiaa, joka vapautuu tai sitoutuu kalorimetrissä olevan reaktion aikana.

- **Differentiaalinen pyyhkäisykalorimetri** (engl. *differential scanning calorimeter*).
  Differentiaalinen pyyhkäisykalorimetri mittaa eri materiaaleja ja vertailee tuloksia keskenään.

- **Isoterminen titraus kalorimetri** (engl. *Isothermal titration calorimeter*).
  Reaktiolämpöä tutkitaan kemiallista menetelmää, titrausta, varten. Tämän tyyppistä kalorimetriä käytetään yleensä lääketeollisuudessa.

- **Röntgensäteily mikrokalorimetri** (engl. *X-ray microcalorimeter*).
  Röntgensäteily mikrokalorimetri on mikrokalorimetri, joka tutkii röntgensäteilyä, ja sen toiminnasta aiheuttavia lämpömuutoksia materiaaleissa. [21]

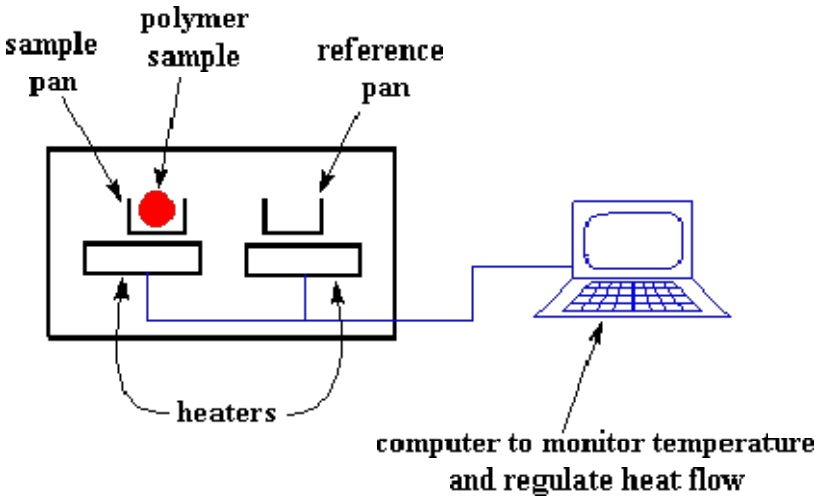
5.2 DSC kalorimetri

DSC kalorimetri, eli differentiaalinen pyyhkäisykalorimetri (engl. *Differential scanning calorimeter*) on kalorimetri, jonka toiminnan perusidea on materiaalien ominaisuuksien vertailu keskenään. Laajalla lämpötila-alueella lämmön vaikutukset voidaan nopeasti tunnistaa käyttäen DSC-kalorimetria [14]. Mittauksen avulla voidaan laskea lämpökapasiteetti ja lämmönsiirto sekä piirtää faasidiagrammi [14].

Toisin sanoen differentiaalinen pyyhkäisykalorimetri tarkoittaa tutkittavan kohteen ja referenssin lämpövirran eron muutoksen mittausta. Se onnistuu ohjelman avulla, joka ohjaa lämpötilan muutosta. Se tarkoittaa sitä, että ohjelma on osa kalorimetria, ja se määrittää mittauksen toimintatavan. [14]

Kuitenkin on muistettava, että lämpö siirtyy vain ja ainoastaan, jos on olemassa lämpötilaero.

Kuvassa 10 on esitetty DSC-kalorimetrin rakenne ja toiminta.



Kuva 10. *DSC rakenne ja perusperiaate [25]*

Kuvassa näkyy, että DSC-kalorimetrissä on kaksi pannua, joita lämmitetään. Toisessa pannussa on tutkittava materiaali, ja toisessa on referenssi materiaali. Tietokoneen avulla ohjataan pannujen lämmitystä ja saadaan dataa materiaalien mittauksista.

5.3 Ongelma

Aiemmin tutustuttiin kalorimetriin, sen tyyppeihin ja toimintaan. Ennen sitä saatiin käsitys faasimuutosmateriaaleista. Tässä diplomityössä käytetään PCM-materiaalia tekstiilialalta, koska Tampereen teknillisen yliopiston Materiaalitekniikan laitoksella tutkitaan ja käsitellään niitä.

Tällä hetkellä ei ole olemassa standardisoitua tapaa mitata PCM-materiaaleja. Aiemmin mainittiin, että tekstiilialalla käytetään faasimuutosmateriaaleja, jotka voivat olla hyödyllisiä ihmisten vaatteissa. Tällä hetkellä on jo olemassa joitakin tekstiilisiä PCM-materiaaleja. Sen takia tarvitaan standardi mittaustapa PCM-materiaaleille, jonka ansiosta valmistajat voisivat arvioida käyttämiensä faasimuutosmateriaalien ominaisuuksia ja mahdollisuuksia, ja pystyisivät täten kertomaan tuotteistaan.

Faasimuutosmateriaaleja varten toteutettu kalorimetrin mahdollistaa sen, että kaikki valmistajat voisivat tehdä samoja PCM-materiaalien mittauksia. Tällä tavalla, he voisivat saada vertailukelpoisia tuloksia. Vielä ei ole päätetty, millainen kalorimetrin pitäisi olla, joten tässä työssä suunnitellaan ja toteutetaan DSC-kalorimetri, jota olisi helppo ja hyödyllistä käyttää faasimuutosmateriaalien määrittelemisessä.

Tässä diplomityössä valittiin DSC-kalorimetri sen takia, että valmistajilla olisi mahdollisuus vertailla heidän käyttämiään tekstiilejä, joihin on yhdistetty PCM-materiaalia, ja tavallisia tekstiilejä.

5.3.1 Kalorimetrin vaatimukset

Faasimuutosmateriaaleja varten suunnitellun DSC-kalorimetrin pitäisi toteuttaa seuraavat määrittelyt:

- **Lämmön sitoutuminen**.
  Kalorimetrin päätehtävä on PCM-materiaalien lämmön sitoutumisen mittaus. Kalorimetrin mittaustulosten perusteella voidaan laskea, kuinka faasimuutosmateriaali voi sitoa itseensä energiaa ja kuinka paljon se pystyy vapauttamaan siitä energiaa joissain olosuhteissa.

- **Lämpötilan muutos**.
  Kalorimetrin on pystyttävä lämmittämään testattavaa materiaalia tarpeeksi käyttämällä ulkoista virtalähdettä, noin 20 °C:sta 60 °C:seen.

- **Lämpötilan luku**.
  Laite mahdollistaa lämpötilan lukemisen materiaalin molemmilta puolilta.

- **Lämpötilan tarkkuus**.
  Tutkittava materiaali voi lämmetä epätasaisesti ja sitä voidaan myös lämmittää epätasaisesti. Tämän kaltaisia virheitä voidaan välttää tai mahdollisesti vähentää käyttämällä useita mittauspisteitä ja tekemällä tarkkoja lämpömittauksia 0,1 °C tarkkuudella.

- **Eristävyys**.
  Tutkittava materiaali pitää eristää ympäristötekijöiltä, jotka voivat vaikuttaa materiaalin lämmittämiseen ja lämpötilan lukemiseen.

- **Käytettävyys**.
  Järjestelmän tulee sisältää helppokäyttöinen fyysinen laite (varsinainen kalorimetri) ja PC-ohjelma, jonka avulla mittaustuloksia voidaan katsella ja tutkia.

- **Turvallisuus**.
  Järjestelmän pitää pystyä tallentamaan kaikki tulokset kahteen paikkaan, sekä tietokoneen sisäiselle asemalevylle, että ulkoiselle muistitikulle.

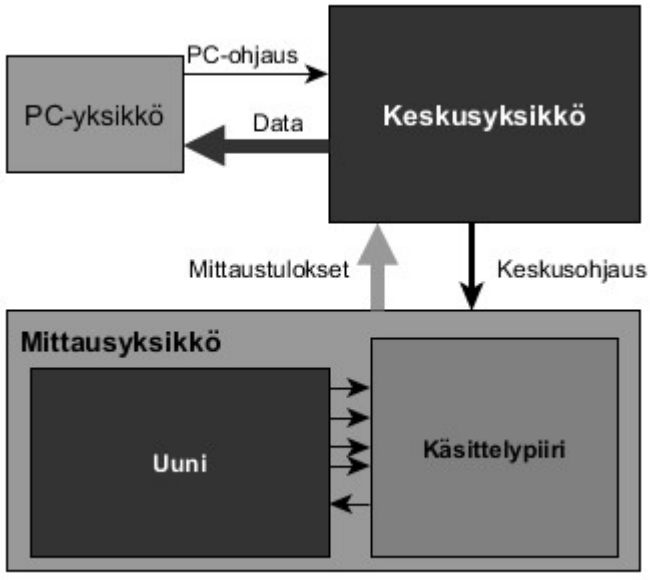
- **Luotettavuus**.
  Järjestelmän tulee olla pienikokoinen, eli kompakti mutta se ei saa olla herkkä, jotta sitä voidaan käsitellä useita kertoja.

5.4 Järjestelmän rakenteellinen kuvaus

Kokonaisuudessaan kalorimetrin järjestelmä koostuu kolmesta moduulista tai yksiköstä, ja muutamista väylistä:

- PC-yksikkö
- Keskusyksikkö
- Mittausyksikkö
- Väylä PC-yksikön ja keskusyksikön välillä
- Väylä keskusyksikön ja mittausyksikön välillä

Kuvassa 11 on esitetty kaikki moduulit ja miten ne kommunikoivat keskenään. Jokaisen yksikön toiminta selitetään myöhemmin. Lisäksi kuvassa näkyy se, mitkä ovat väylät moduuleiden välillä, mutta miten ne on toteutettu, selitetään seuraavassa luvussa, Kalorimetrin toteutus.



Kuva 11. *Järjestelmän lohkokaavio*

5.4.1 PC-yksikkö

PC-yksikön tärkein ominaisuus on se, että käyttäjä toimii enimmäkseen sen kanssa ja vähemmän muiden yksiköiden kanssa. PC-yksikön tehtävien ja vaatimuksien lista on esitetty alla. PC-yksikkö edustaa tietokonetta, mutta paremmin voisi sanoa, että se edustaa tietokoneessa olevaa ohjelmaa.

PC-yksikön vaatimukset ja toiminnat.

- **Keskusyksikön käynnistys**.
  Edellisessä kappaleessa sanotaan, että käyttäjä toimii tämän yksikön kanssa, joten PC-yksikön on käynnistettävä keskusyksikkö.

- **Keskusyksikön ohjaus** ja **asetus**.
  Käyttäjä pystyy asettamaan oikeiksi ja halutuiksi keskusyksikön toiminnat.

- **Käytettävyys**.
  Ohjelman on toimittava tavallisella käyttöjärjestelmällä kuten Windows. Sen takia sitä varten valittiin Microsoftin ohjelmakehitysympäristö Microsoft Visual Studio. Tällä ympäristöllä voidaan toteuttaa Windows-sovelluksia, jotka ovat helppokäyttöisiä tavallisille käyttäjille. Ohjelmointikieleksi valittiin C# (C sharp), joka sopii erinomaisesti .NET frameworkin kanssa yhteen.

- **Mittaustulosten esitys** ja **tallennus**.
  Keskusyksiköstä saapuvia mittaustuloksia tulostetaan ruudulle, josta ne voidaan tallentaa.

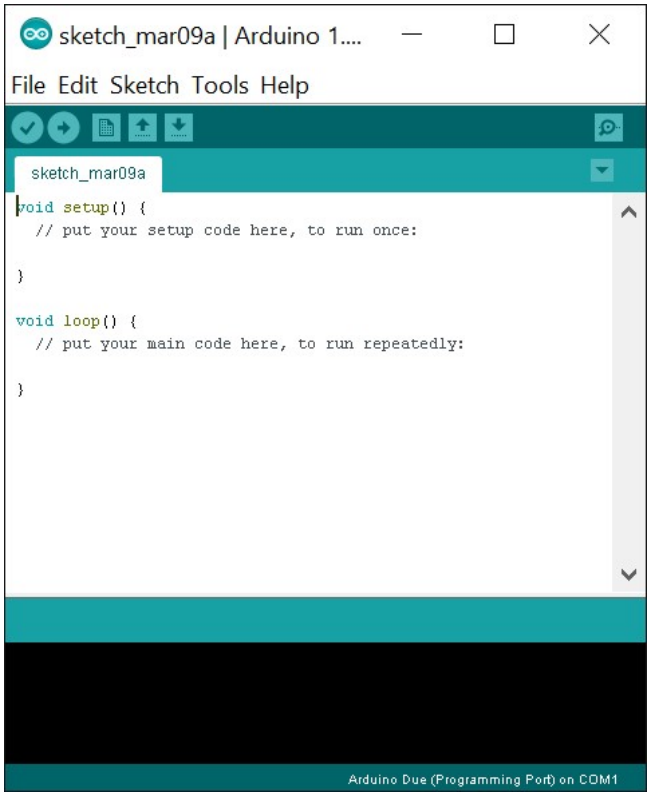
5.4.2 Keskusyksikkö

Järjestelmän ytimessä toimii ns. keskusyksikkö, joka niin kuin kuvassa 11 näkyy, ensisijaisesti yhdistää PC-yksikön ja mittausyksikön.

Keskusyksikkönä toimii Arduino. Se on alusta, joka koostuu mikro-ohjaimesta (mikrokontrollerista), HW-alustasta ja ohjelmointiympäristöstä [2].

Arduino perustuu 8-bittisiin Atmel AVR-mikrokontrollereihin. Arduino Uno:n tapauksessa mikrokontrollerina toimii Atmel-prosessori ATmega328.

Keskusyksikön alustaksi oli valittu Arduino, koska se käyttää ohjelmointikielenä C/C++ kieltä. Sen lisäksi, Arduino-piirilevy on aina valmiiksi koottu komponentteineen, joten sitä tarvitsee vain ohjelmoida ja kytkeä muihin ulkoisiin komponentteihin tai piirilevyihin. Arduinon viralliselta Internet-sivulta löytyy paljon vinkkejä sen käyttämiseen [4].



Kuva 12. *Arduino IDE aloitusikkuna*

Arduino tarjoaa oman ilmaisen ohjelmistonkehitysympäristön nimeltään Arduino IDE (engl. *Integrated Development Envinroment*), kuva 12. Tätä ohjelmistoa voidaan ladata viralliselta nettisivulta. Arduinon käynnistämiseen tarvitaan USB-kaapeli, jonka kautta piiri saa käyttöjännitteen, ja jonka avulla sitä pystyy ohjelmoimaan käyttäen Arduino IDE ohjelmistoa.

Keskusyksikön vaatimukset ja toiminnat:

- Keskusyksikkö saa ohjauksen ja asetukset PC-yksiköltä ja toimii niiden mukaisesti.
- Ohjaa mittausyksikköä ja asettaa sen komponenttien parametrit halutuksi.
- Keskusyksikkö saa dataa mittausyksiköltä, sekä muuntaa ja käsittelee sitä.
- Lähettää dataa PC-yksikköön tulevaa käsittelyä varten.

Tarkemmin keskusyksikön toimintaa esitellään tässä diplomityössä myöhemmin.

5.4.3 Mittausyksikkö

Viimeinen kolmas moduuli on mittausyksikkö, joka on esitetty kuvassa 11. Sen tulee toteuttaa monia tehtäviä:

- Mittausyksikkö saa ohjauksen ja asetukset keskusyksiköltä.
- Asettaa kaikki komponenttien parametrit asetusten mukaisesti.
- Pystyy lämmittämään tutkittavaa materiaalia tehokkaasti.
- Mahdollistaa lämpötilan ja sen muutosten lukemisen.
- Mittausyksikkö saa dataa omilta antureilta, joita tulee olla vähintään kuusi tarkkaa mittausta varten.

Mittausyksikköä varten pitää piirtää kytkentäkaavio, valmistaa piirilevy ja juottaa komponentit. Yksikön toteutuksesta ja komponenttivalinnoista kerrotaan seuraavassa luvussa.

6. KALORIMETRIN TOTEUTUS

Luvussa 5 tutustuttiin kalorimetri järjestelmän rakenteelliseen kuvaukseen. Kalorimetrissä on kolme moduulia: PC-yksikkö, keskusyksikkö ja mittausyksikkö. Niistä ja niiden toteuttamisesta kerrotaan tässä luvussa.

6.1 Väylät

Edellisessä luvussa mainittiin, että kaikki moduulit kommunikoivat keskenään käyttäen eri tiedonsiirtoväyliä.

Väylä on ns. alijärjestelmä, jonka tärkein tehtävä on tiedon siirto paikasta toiseen. Tässä tapauksessa tarvitaan kaksi tiedonsiirtoväylää: PC-yksikön ja keskusyksikön väliin ja keskusyksikön ja mittausyksikön väliin.

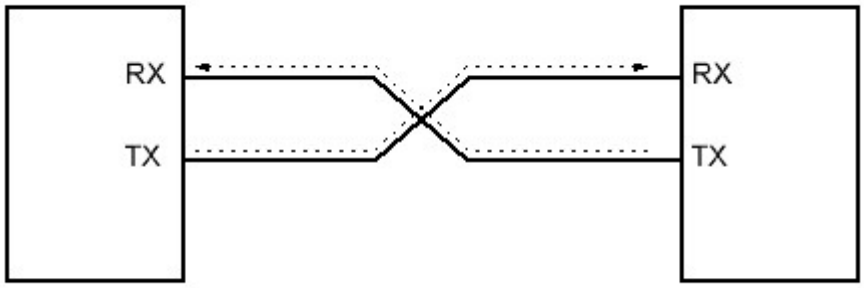
Arduino Uno alusta tarjoaa useita sarjaliikenteitä kuten UART ja SPI, niistä ja niiden käytöstä kerrotaan nyt.

6.1.1 PC-yksikön ja Keskusyksikön välillä tiedonsiirto

Keskusyksikköä edustaa ohjelmoitu Arduino-piiri, joka on yhdistetty PC-yksikköön fyysisesti USB-liittimen kautta. Se voi toimia sekä piirin käyttöjännitteenä, että tiedonsiirronväylänä.

Mikrokontrolleri, joka sisältyy Arduino-piiriin, on ATmega328. Arduinon datalehden mukaan [6] ATmega328 mahdollistaa UART-tiedonsiirron, jota yleensä käytetään USB-protokollan yhteydessä. Perinteisen UART:n (engl. *Universal Asynchronous Receiver-Transmitter*) sijaan Atmel mikrokontrollereissa käytetään parannettua versiota, eli USART rajapintaa (engl. *Universal Synchronous/Asynchronous Receiver-Transmitter*).

USART on sarjaliikenne, eli siinä on yksi johdin yhteen suuntaan. Sen kommunikaatio tapahtuu käyttämällä pinnejä RX (*Receiver*, vastaanottaja) ja TX (*Transmitter*, lähettäjä) USB-liittimen avulla [3]. Kuvassa 13 näkyy, miten kahden laitteen pinnit kytketään yhteen, mutta ristiin.



Kuva 13. *UART rajapinnan kytkentä*

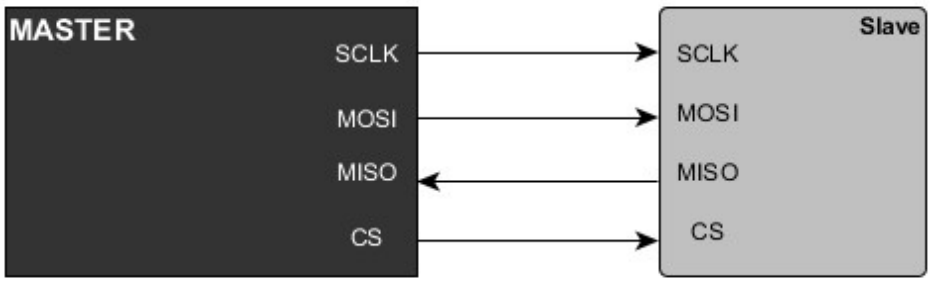
USB-protokolla ei ole yhteensopiva UART:n kanssa, jonka takia Arduino-piirissä on oma muunnin sitä varten.

6.1.2 Keskusyksikön ja Mittausyksikön välillä tiedonsiirto

Keskusyksikön ja mittausyksikön välillä tiedonsiirron muotona ei voida käyttää muuta kuin UART-rajapintaa. Toinen yleinen sarjaliikennemuoto SPI (eng. *Serial Peripheral Interface*) valittiin, koska mittayksikössä olevat maxim-piirit käyttävät niitä.

SPI on synkroninen sarjaliikennemuoto, eli sillä on aina erillinen kellosignaali. Kommunikaatiossa on ainakin kaksi osapuolta, ne ovat isäntä (*master*) ja orja (*slave*). SPI-tiedonsiirrossa aina toinen on master ja toinen on slave, jolloin niiden roolit eivät sekoitu. Tämän takia kellosignaali on tahdistettu Master:n tuottamaan kellosignaaliin [15].

Kuvassa 14 näkyy, miten SPI-signaalit yhdistetään kahden osapuolen välillä.



Kuva 14. *SPI yhteys kahden osapuolen välillä*

SPI-tiedonsiirrossa on seuraavat signaalit.

- **SCLK**, (engl. *Serial Clock*).
  Kellosignaali, joka lähtee isäntälaitteelta orjalaitteelle

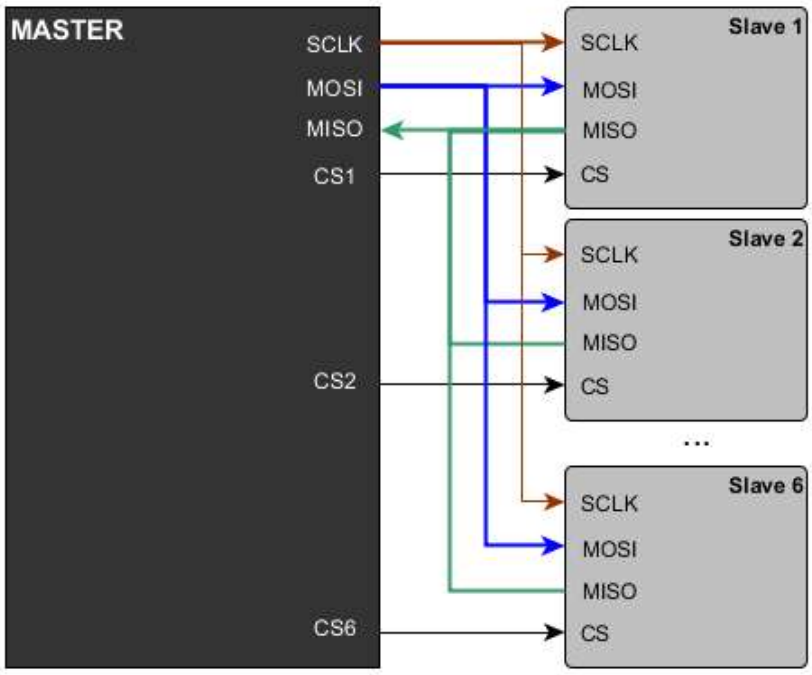
- **MOSI**, (engl. *Master Out Slave In*).
  Data siirtyy isäntälaitteelta orjalaitteelle

- **MISO**, (engl. *Master In Slave Out*).
  Data siirtyy orjalaitteelta isäntälaitteelle

- **CS**, (engl. C*hip select*).
  CS on osoitussignaali, jolla annetaan käsky orjalaitteelle, että sen kanssa tullaan kommunikoimaan.

Työssä käytössä oleva Arduino-piiri mahdollistaa UART:n lisäksi SPI-tiedonsiirron käytön samaan aikaan. Työn mittayksikössä käytetään useita lämpötila-antureita ja sen takia useita maxim-piirejä, joten SPI-kytkentä näyttää tällöin erilaiselta, koska käytössä on enemmän kuin yksi orjalaite. Kuvassa 15 on esitetty, miten kommunikaatio onnistuu, kun käytetään vain yhtä isäntälaitetta ja kuutta orjalaitetta, joiden kanssa kommunikoidaan.

Tällöin isäntälaitteella pitää olla kuusi Chip Select signaalia, joita ohjaten Master pystyy toimimaan kaikkien kuuden orjalaitteen kanssa.



Kuva 15. *SPI kytkentä kuutta orjalaitetta varten*

6.2 PC-yksikön toteutus

Tässä luvussa kerrotaan siitä, mitkä vaatimukset oli asetettu suunnittelulle, ja miten ne pystyttiin toteuttamaan.

- **Keskusyksikön käynnistys**:

PC-yksikkö pystyy muodostamaan yhteyden mittausyksikköön sarjaliikenteen avulla. Sitä varten käyttäjän on valittava COM portti ja nopeus. Tämän jälkeen ohjelma itse lähettää käskyjä mittausyksikköön ja alkaa kommunikoida sen kanssa. Mittauksen jälkeen ohjelma lopettaa mittausyksikön toiminnan.

- **Keskusyksikön ohjaus** ja **asetus**:

PC-yksikön on osattava monia tehtäviä, jotka ovat esitetty alla olevassa listassa:

- Lämpömittaus
- Datan tallennus
- Datan lataus
- Kalibroinnin suorittaminen
- Kalibroinnin tulosten käyttö

Kolmesta ensimmäisestä tehtävästä kerrotaan aikaisemmin tekstissä tai tämän kohdan jälkeen. Tässä esitetään kalibroinnin toteutus.

Ohjelman tärkeä ominaisuus on kalibroinnin toteuttaminen. Ohjelma pystyy tutkimaan antureiden arvoja, kun anturit ovat samassa paikassa ja lämmössä. Käyttäjän on valmistettava tasapainoinen ja ympäristötekijöiltä suojattu paikka. Sen jälkeen arvot voidaan tallentaa, jotta niitä pystytään hyödyntämään antureiden mahdollisten virheiden välttämiseksi tai ainakin vähentämiseksi.

- **Käytettävyys**:

Ohjelma edustaa Windows Form sovellusta, joten sitä on helppo käyttää. Tästä kerrotaan lisää myöhemmin.

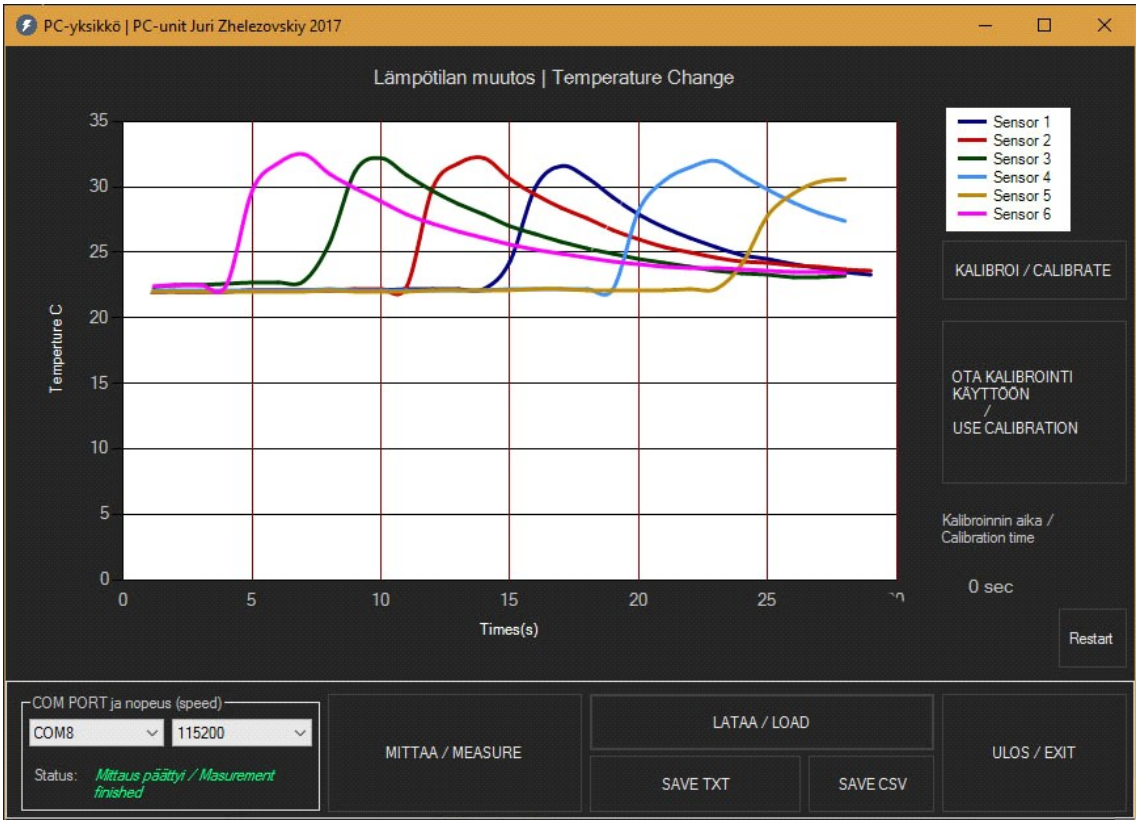
- **Mittaustulosten esitys** ja **tallennus**:

Mittauksen käynnistyttyä ohjelma tulostaa reaaliajassa saapuvia mittaustuloksia ruudulle. Mittauksen päätyttyä tuloksia voidaan tarkkailla ja tutkia skaalaamalla käyriä.

Mittaustulokset voidaan tarvittaessa tallentaa teksti-tiedostona. Ohjelma itse tallentaa tulokset siinä muodossa, jotta näitä tiedostoja voidaan myöhemmin avata katsottavaksi. Lisäksi mittaustulokset voidaan tallentaa CSV muodossa, jotta niitä pystytäisiin avaamaan Excel ohjelmistolla.

6.2.1 Ohjelmisto

PC-yksikön ohjelman koodaaminen vaati 1017 riviä. Koko koodi on esitetty liitteessä F. Ohjelma on kirjoitettu C#-kielellä Microsoft Visual Studio ympäristössä. Se on .NET Framework, joka on Microsoftin toteuttama ohjelmistokomponenttikirjasto. Ohjelma on Windows Form sovellus, joten sitä voidaan helposti käyttää Windows käyttöjärjestelmässä. Kuvassa 16 on esitetty PC-yksikön toteutus.



Kuva 16. *PC-yksikön toteutus*

Kalibroinnin aikana ohjelma tarkastelee antureiden arvojen eroja ja tallentaa ne. Käyttäjä voi ottaa kalibroinnin käyttöön, tällöin mittauksen aikana kalibroinnin tulokset otetaan huomioon laskelmissa.

6.3 Keskusyksikön toteutus

Aikaisemmin mainittiin, että keskusyksikkönä toimii Arduino-piiri, kuvassa 17, jota tullaan ohjelmoimaan.

Kuva 17. *Arduino Uno – piiri*

Ohjelmointi tapahtuu Arduino Ide ympäristössä C-kielellä.

Keskusyksikkö kommunikoi PC-yksikön kanssa UART-tiedonsiirtomuodon avulla, ja mittausyksikön SPI-tiedonsiirtomuodon avulla, joista molemmat ovat sarjamuotoisia.

6.3.1 Ohjelmisto

Keskusyksikön ohjelman koodaaminen vaati 564 riviä. Tässä luvussa käydään läpi lyhyesti, miten ohjelma on toteutettu. Koko koodi on esitetty liitteessä E. Seuraavaksi esitellään vaatimukset sekä ohjelmalliset ratkaisut niille.

- **Keskusyksikkö saa ohjausta ja asetukset PC-yksiköstä ja toimii niiden mukaisesti.**

Vaatimuksien mukaan Keskusyksikön on toimittava PC-yksikön kanssa käyttäen UART tiedonsiirtoa. Arduino-piireissä ohjelmallisesti se tapahtuu yksinkertaisesti.

```
Serial.begin( 115200 );
```

Sulkuihin on kirjoitettu nopeus, joka pitää muistaa. PC-yksikössä olevaa ohjelmaa yhdistetään tähän ohjelmaan käyttämällä samaa nopeutta. Tästä kerrotaan tarkemmin PC-yksikön toteutus osassa.

Seuraavaksi ohjelman tehtävä on odottaa käskyä PC-yksiköstä.

```
while ( Serial.available() == 0 );
  char letter = Serial.read();
  // wait for symbol to start
  if( letter == '1' )
  {
  …… // varsinainen ohjelma
  } // if letter == 1
```

Ohjelma odottaa dataa, ja kun PC-yksiköstä tulee sarjaliikennettä pitkin ”1”, silloin ohjelma käynnistää varsinaisen toimintansa.

- **Ohjaa Mittausyksikköä ja asettaa sen komponentit halutuksi.**

Käytössä on kuusi anturia ja niitä varten kuusi MAX-piiriä. Jokaista piiriä varten on oman luokan olio. Jokaisella anturilla on oma Chip Select pinni, jotta niitä olisi mahdollista ohjata erikseen.

```
// luodaan luokan olioita
// create objects of class
MAX31865_piiri sensor1( MAX31865_piiri( CS_PIN_1 ) );
MAX31865_piiri sensor2( MAX31865_piiri( CS_PIN_2 ) );
MAX31865_piiri sensor3( MAX31865_piiri( CS_PIN_3 ) );
MAX31865_piiri sensor4( MAX31865_piiri( CS_PIN_4 ) );
MAX31865_piiri sensor5( MAX31865_piiri( CS_PIN_5 ) );
MAX31865_piiri sensor6( MAX31865_piiri( CS_PIN_6 ) );
```

Komponenttien asetus-käsky toteutetaan seuraavasti:

```
  sensor1.asetukset(  );
  sensor2.asetukset(  );
  sensor3.asetukset(  );
  sensor4.asetukset(  );
  sensor5.asetukset(  );
  sensor6.asetukset(  );
```

Suoritetaan funktio nimeltään asetukset.

```
void MAX31865_piiri::asetukset( void )
{
    uint8_t asetus_bitit = 0;
    /**
     * 0xC3 = 0b 1100 0011
     * D7 = 1: Vbias 1:on 0:off
     * D6 = 1: Conversion mode 1:Auto 0:off
     * D5 = 0: 1-shot 1:on 0:off
     * D4 = 0: 1:3 Wire, 0:2/4 Wire
     * D3, D2: Fault detection
     * D1 = 1: Fault status 1:true
     * D0 = 1: 1:50Hz, 0:60Hz
    */
    asetus_bitit = 0xC3;
```

```
    // send settings to MAX
    digitalWrite( this->cs_pin, LOW );
    // lähetetään konfigurointi asetukset, koska se on write to MAX,
osoite on 0x8h, sivu 13
    SPI.transfer( 0x80 );
    SPI.transfer( asetus_bitit );
    digitalWrite( this->cs_pin, HIGH );
}
```

Tässä lähetetään bitit MAX-piireille, joissa bitit toimivat asetusarvoina. Tässä käytetään nelipistemenetelmä.

- **Keskusyksikkö saa dataa Mittausyksiköstä, muuntaa ja käsittelee sitä.**

```
sensor1.lue_kaikki( );
```

Funktio ”lue_kaikki” on niin pitkä, että sen toteutus on esitetty liitteessä F.

- **Se lähettää dataa PC-yksikköön tulevaa käsittelyä varten.**

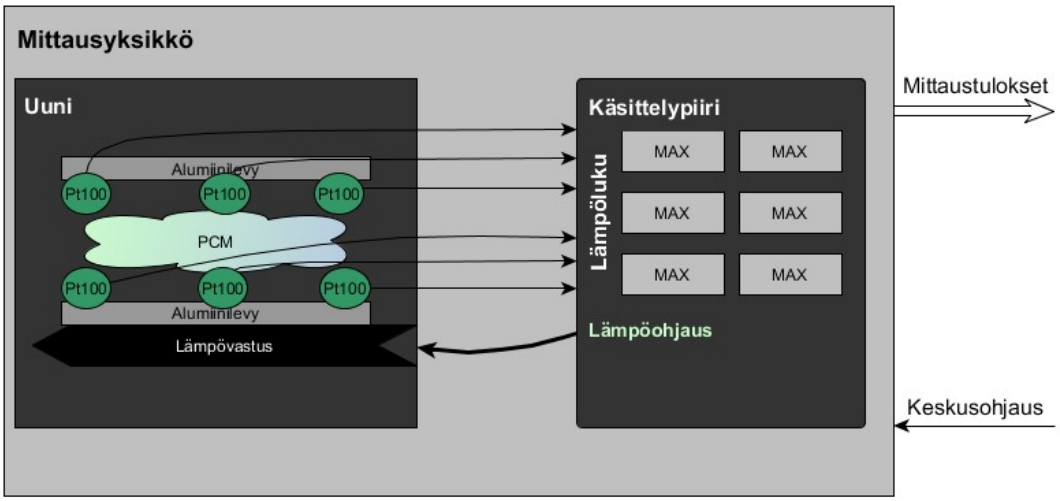
Datan lähettäminen taas tapahtuu yksinkertaisesti UART-sarjaliikennettä käyttäen. Sitä lähetetään PC-yksikköön, jonka on vastaanotettava sitä tietoa.

```
Serial.print( " T4 = ");
Serial.print( temperature );
Serial.println(" C" );
```

6.4 Mittausyksikön toteutus

Vaatimusten perusteella mittausyksikkö päätettiin toteuttaa jakamalla se kahteen osaan. Toinen osa kommunikoi keskusyksikön kanssa ja toinen suorittaa mittaukset.

Kuvassa 18 on esitetty mittausyksikön yksityiskohtainen lohkokaavio.



Kuva 18. *Mittausyksikön lohkokaavio*

Itsestään mittausyksikkö koostuu käsittelypiiristä ja uunista.

Uunissa tapahtuu varsinainen lämmönmittaus. Käsittelypiiri saa ohjaukset keskusyksiköltä ja se asettaa ensin omat sisällään olevat komponentit asetusten mukaisesti. Sitten käsittelypiiri odottaa aloituskäskyä keskusyksiköltä. Kun käsky tulee, käsittelypiiri ohjaa uunissa olevaa lämpövastusta.

Uunin keskipisteessä on tutkittava PCM-materiaali. Sitä tutkii kuusi PT-100 anturia, jotka ovat sen molemmin puolin. Lämpöanturit on laitettu faasimuutosmateriaaliin kiinni kahden alumiinilevyn avulla.

Kun lämpövastus lämpenee, se lämmittää yllä olevaa alumiinilevyä. Näin ollen lämpöanturit mittaavat tuotettua lämpöä. Toisella puolella anturit tutkivat PCM-materiaalin läpi menevää lämpöä. Erotuksen avulla voidaan laskea, kuinka paljon energiaa on sitoutunut faasimuutosmateriaaliin.

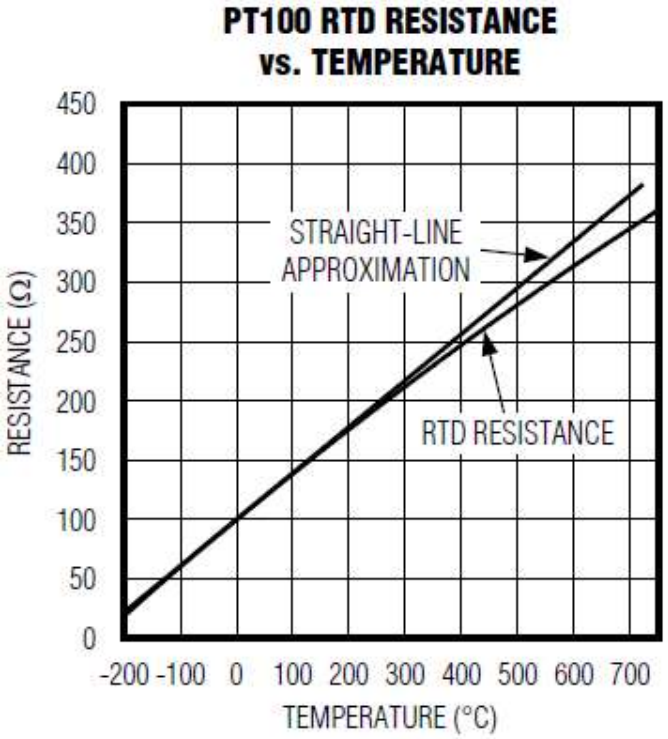
Jokaista lämpöanturia kohden on MAX-mikropiiri, joista selitetään myöhemmissä luvuissa.

6.4.1 Lämpöanturin valinta

Lämpöanturin valinnalla on erittäin tärkeä rooli, koska tässä diplomityössä se suorittaa erittäin tarkkoja mittauksia.

Erinomaiseksi ratkaisuksi osoittautui lämpöanturi PT100. Se kuuluu RTD-antureiden joukkoon (engl. *resistance temperature detector*). Lyhyesti sanottuna, se on vastus, jonka resistanssi riippuu ympäristön lämpötilasta.

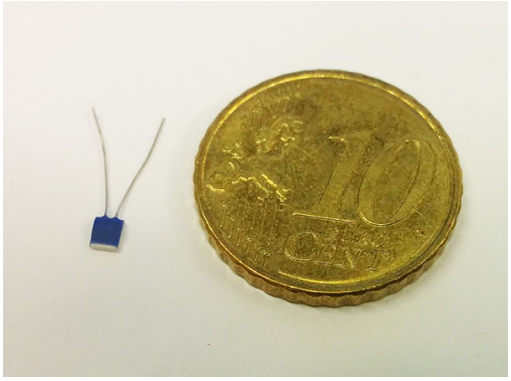
PT100 nimi johtuu anturin materiaalista (platina) ja vastuksen arvosta 100 Ω. Anturin resistanssi on tasan 100 Ω lämpötilan ollessa 0 °C. Platina on erittäin tarkka johtava materiaali. Sen takia sitä käytetään paljon tarkoissa antureissa.



Kuva 19. *PT100:n resistanssin riippuvuus lämpötilasta [20]*

Kuvassa 19 on esitetty anturin resistanssin riippuvuus lämpötilasta. Resistanssi muuttuu lähes lineaarisesti lämpötilan muuttuessa.

PT100 datalehtien mukaan [17] [13] antureilla on muutama tarkkuusluokka, joilla on omat mahdolliset toleranssit. Tässä diplomityössä on käytössä anturit (kuva 20), joiden tarkkuusluokka on A, jonka toleranssi on 0.15 °C [17] [13].



Kuva 20. *Pt100 anturin koko verrattuna 10 sentin kolikkoon*

Anturi on erittäin pieni ja herkkä, joten se sopii erinomaisesti tähän tehtävään, mutta toisaalta se tuottaa käsittelyyn liittyviä vaikeuksia.

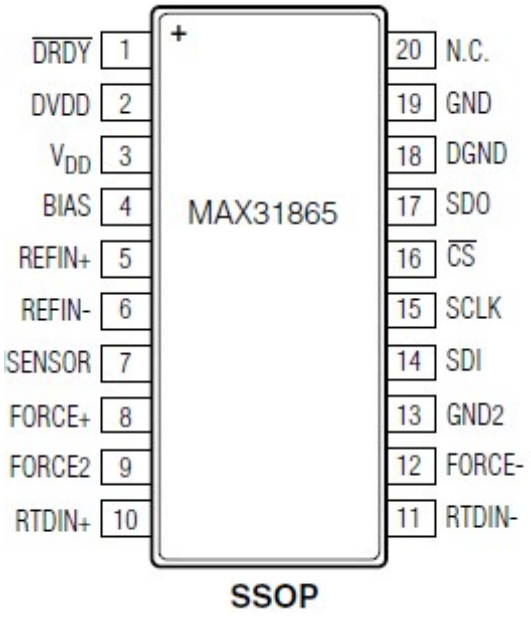
6.4.2 Mikropiirin tarve ja valinta

PT100 on erittäin herkkä ja tarkka anturi. Toisaalta se toteuttaa erinomaisesti asetetut vaatimukset, mutta toisaalta se vaatii tarkan mittauselektroniikan. Lämpömittausta varten tarvitaan hyvin tasainen referenssijännite, jota keskusyksikössä oleva Arduino-piiri ei pysty tuottamaan. Itse asiassa Arduino pystyy tuottamaan tarvittavan jännitteen, mutta se ei pysty pitämään sitä tarpeeksi tasaisena ja muuttumattomana.

PT100-antureita varten on olemassa mikropiiri valmistajalta MAXIM. MAX31865[20] on RTD-vastuksiin erikoistunut mikropiiri, jonka pääominaisuus on muunnos resistanssista digitaaliseksi (*resistance-to-digital*).

Mikropiiri käyttää SPI-väylää kommunikointia varten, jonka takia mittausyksikkö ja keskusyksikkö on yhdistetty SPI-väylän avulla.

Arduino-piirillä on 10-bittinen AD muunnos [1], mutta MAX-piirillä se on 15 bittiä [20]. Se mahdollistaa resoluution 0.03°C ja maksimivirheen 0,5 °C. Virrankulutus on 3,5mA. Käyttöjännite on 3,0V ... 3,6V, joka sopii hyvin, koska Arduino tarjoaa Vout = 3,3V. Se suojaa herkkiä PT100-antureita ylijännitteeltä. [20]



Kuva 21. *MAX31865 pinnijärjestys*

Kuvassa 21 on esitetty tässä diplomityössä toimiva MAX-piiri, joka palauttaa tulokset digitaalisessa muodossa. Piiriä tulee alustaa ohjelmallisesti ja siitä kerrotaan tässä diplomityössä myöhemmin.

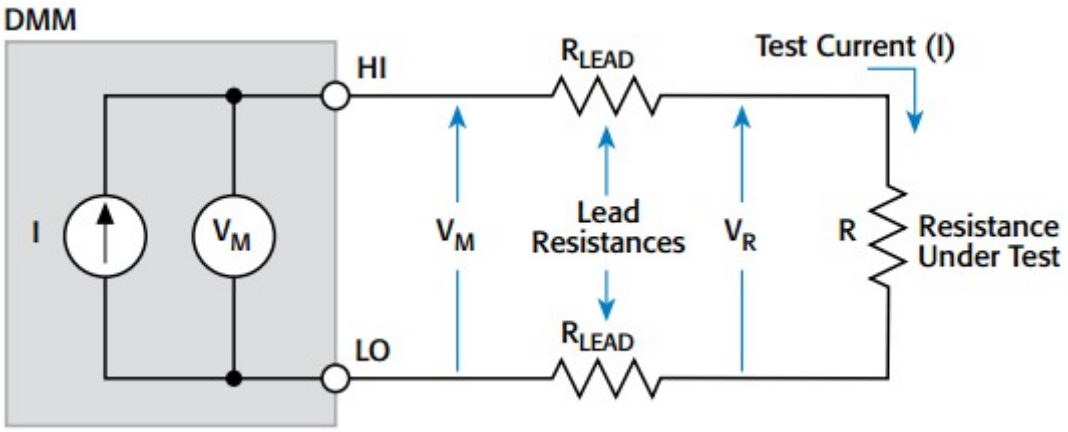
6.4.3 Mikropiirin ja anturin kytkentä

MAX31865 ja PT100 voidaan kytkeä toisiinsa käyttäen kolmea kytkentää [20]:

- kaksipistemenetelmä
- kolmipistemenetelmä
- nelipistemenetelmä

MAX-piirin on mitattava hyvin PT100:n resistanssia. Kolmipistemenetelmä on kaksi- ja nelipistemenetelmien yhdistelmä, joten tässä siitä ei kerrota.

- Kaksipistemenetelmä, kuva 22.



Kuva 22. *Kaksipistemenetelmä [16]*

Vasemmalla puolella on mikropiirin komponentit: virtalähde I ja volttimittari Vm. Oikealla puolella on kaksi johtoa, joilla on resistanssit Rlead ja RUT (*resistance under test*), eli tässä tapauksessa se on PT100.

$V_m \neq V_R$, koska $V_m = V_{Rlead} + V_R + V_{Rlead}$, jolloin

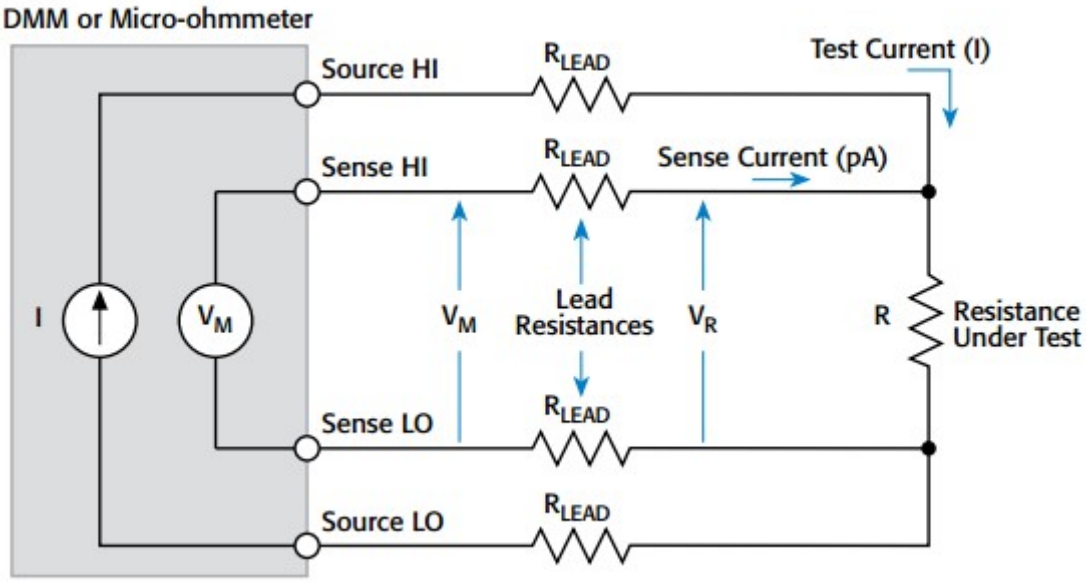
$$\text{Mitattava PT100} = \frac{V_m}{I_{test}} = \frac{V_{Rlead} + V_R + V_{Rlead}}{I_{test}} =$$

$$\frac{I_{test} \cdot R_{lead} + R \cdot I_{test} + I_{test} \cdot R_{lead}}{I_{test}} = R_{lead} + R + R_{lead}$$

$$\text{Mitattava PT100} = 2R_{lead} + R$$

Yhtälöstä näkyy, että mitattavan PT100:n resistanssi sisältää ylimääräiset johtojen resistanssit.

- Nelipistemenetelmä, kuva 23.



Kuva 23. *Nelipistemenetelmä [16]*

Tarkempaa mittausta varten käytetään nelipistemenetelmää.

Volttimittarin ulostulon resistanssi on erittäin suuri, joten I_{sense} on lähes nolla. Sen takia $V_m = V_R$

$$\text{Mitattava PT100} = \frac{V_m}{I_{test}} = \frac{V_R}{I_{test}}$$

Täten mittaus ei riipu mitenkään johtojen resistansseista.

6.4.4 Lämpöluku

MAX-piiri mahdollistaa kaksi tapaa mitata lämpötilaa saadun resistanssin avulla.

Luvussa 6.4.1 on kerrottu siitä, että lämpöanturin resistanssi käyttäytyy lähes lineaarisesti lämpötilan muuttuessa. Näin ollen, lämpötila voidaan laskea käyttäen kaavaa 6.4.4.1.

$$\text{Temperature (°C)} \approx (\text{ ADC code} / 32) - 256 \text{ [20]} \qquad (6.4.4.1)$$

Tarkkaa mittausta varten datalehdessä ehdotetaan käytettäväksi Callendar-Van Dusen kaavaa.

$$R(T) = R_0 \cdot (1 + aT + bT^2 + c \cdot (T - 100)T^3) \text{ [20]}$$

T – nykyinen lämpötila.

$R(T)$ – resistanssi lämpötilan ollessa T.

R_0 – resistanssi lämpötilan ollessa 0℃, meidän tapauksessa se on 100 Ω.

a, b, c ovat kertoimia. IEC 751 standardin mukaan:

a = 3.90830 x 10-3

b = -5.77500 x 10-7

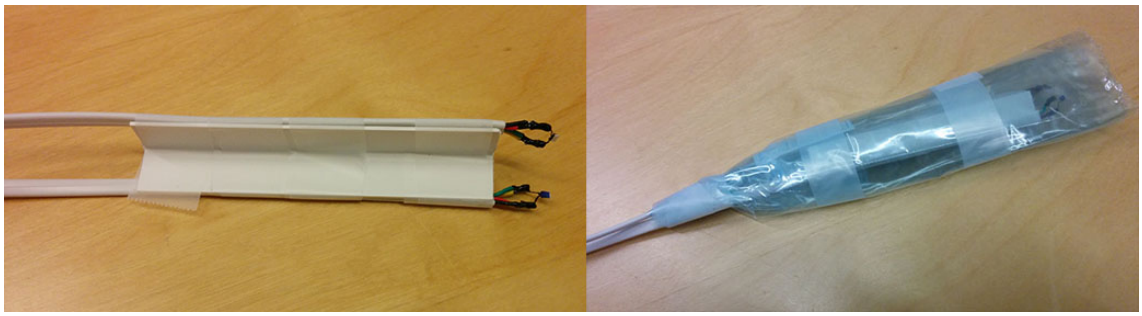
c = -4.18301 x 10-12 jos -200℃ ≤ T ≤ 0 ℃ ja c = 0, jos 0 ℃ ≤ T ≤ 850 ℃

6.4.5 Antureiden kalibrointi

Kalibroinnin tehtävä on poistaa antureiden mahdolliset virheet, jotta ne tuottaisivat samoja arvoja ollessaan samassa lämpötilassa.

PT100-lämpöanturit ovat erittäin tarkkoja ja herkkiä antureita. Lähes kaikki ulkoiset tekijät voivat vaikuttaa niiden toimintaan. Jotta lämpöluku olisi luotettavampi, päätettiin suorittaa antureiden kalibrointi. Sitä varten PC-yksikköön oli lisätty mahdollisuus suorittaa antureiden kalibrointi suoraan, tästä on kerrottu tarkemmin luvussa 6.2 ja 6.2.1.

Ennen kalibrointia anturit eristettiin toisistaan kiinnittämällä ne palkkeihin. Sen jälkeen ne laitettiin vedenpitävästi muovipusseihin. Kuvassa 24 on esitetty eristetyt anturit.



Kuva 24. *Eristetyt anturit*

Muoviastiaan laitettiin jäitä ja kaadettiin kylmää vettä. Tästä koostumuksesta alkoi muodostua jäävesi. Seuraavaksi jääveteen upotettiin kaikki kuusi anturia pareittain. Se näkyy kuvassa 25.

Kuva 25. *Kalibroinnin koe*

Kesti noin puolitoista tuntia, ennen kuin jääveden lämpötila oli lähes nolla. Samaan aikaan käynnistettiin PC-yksikön avulla kalibrointi tehtävä. Se tutki lämpöantureiden arvoja ja tallensi niitä. Antureiden lopulliset arvot on esitetty taulukossa 1. Käyttäjä voi ottaa käyttöön nämä arvot PC-yksikössä, jolloin ohjelma käyttää niitä laskelmissaan.

Taulukko 1. *Kalibroinnin tulokset*

	Lämpötilat, °C		
Kokeen nro / Anturin nro	1	2	3
1	7,9	7,7	0,0
2	7,7	7,4	-0,2
3	7,9	7,8	0,0
4	8,1	7,8	0,0
5	8,0	7,8	0,0
6	8,0	7,8	0,0

Kuten on sanottu, PT100-lämpöanturit ovat erittäin herkkiä, joten lämpötilan lukemisen virheiden vähentämiseksi käytössä on kuusi anturia.

6.4.6 Levyjen lämmitykseen tarvittava teho

PCM-materiaalin lämmitystä varten, pitää tietää kuinka paljon tehoa tarvitaan. Mittausyksikössä alumiinilevy ja teräslämpövastus lämpenevät yhtä aikaa.

Tehoa lasketaan kaavalla $P = \frac{W}{t}$, jossa *W* on työ, tai energia ja *t* aika. Koska energia on tehty työ, sitten teho lasketaan näin $P = \frac{Q}{t}$, jossa Q on energia.

Käytetyn alumiinilevyn ja teräslämpövastuksen ominaislämpökapasiteeteista voidaan laskea energia.

Luvussa 4 esitettiin kaava (4.1), jonka avulla voidaan laskea, kuinka paljon energiaa tarvitaan materiaalin lämmittämiseen tietyllä lämpövälillä.

$$Q = c \cdot m \cdot \Delta T$$

Massa lasketaan seuraavaa kaavaa käyttäen:

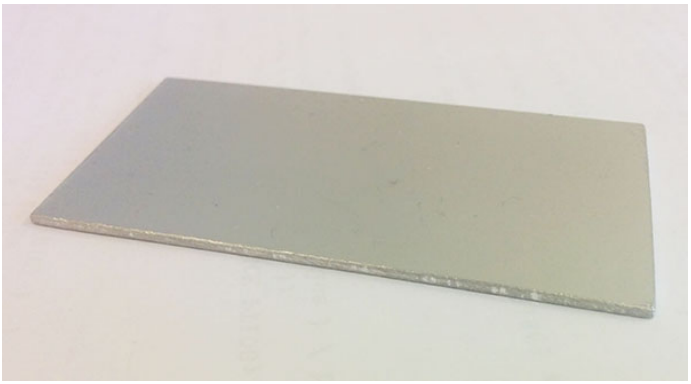
$$m = V \cdot \rho$$

Jossa ρ on tiheys(*kg/m3*) ja V on tilavuus. Alumiinin ja teräksen tiheydet ovat tiettyjä, mutta niiden tilavuuden laskemisessa käytetään toista kaavaa:

$$V = W \cdot H \cdot L$$

Jossa W on leveys, H on korkeus ja L on pituus.

Työssä käytetään alumiinilevyä, joka on esitetty kuvassa 26, ja teräslämpövastusta, joka löytyy kuvasta 27.



Kuva 26. *Alumiinilevy*

Alumiinilevyn ja teräslämpövastuksen koot ovat:

H_1 = 1,5mm H_2 = 1,0mm

W_1 = 40mm W_2 = 40mm

L_1 = 76mm L_2 = 76mm

Alumiinin ja teräksen tiheydet ovat seuraavat:

$\rho_1 = 2700 \frac{kg}{m^3}, \rho_2 = 7820 \frac{kg}{m^3}$ [27]

Alumiinin ja teräksen ominaislämpökapasiteetit ovat:

$c_1 = 897 \frac{J}{K \cdot kg}, c_2 = 490 \frac{J}{K \cdot kg}$ [29]

Lasketaan ensin alumiinilevyn lämpökapasiteetti, C_1

$$C_1 = W_1 \cdot H_1 \cdot L_1 \cdot \rho_1 \cdot c_1 = 0{,}04 \cdot 0{,}0015 \cdot 0{,}076 \cdot 2700 \cdot 897 = 11{,}04\, J/K$$

Seuraavaksi lasketaan teräslämpövastuksen lämpökapasiteetti, C_2

$$C_2 = W_2 \cdot H_2 \cdot L_2 \cdot \rho_2 \cdot c_2 = 0{,}04 \cdot 0{,}001 \cdot 0{,}076 \cdot 7820 \cdot 490 = 11{,}59\, J/K$$

c_1 ja c_2 on alumiinin ja teräksen ominaislämpökapasiteetit.

ρ_1 ja ρ_2 on alumiinin ja teräksen tiheydet.

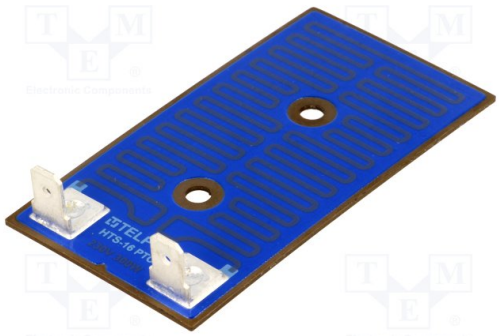
Kokonaislämpökapasiteetti lasketaan summaamalla.

$$C = C_1 + C_2 = 11{,}04 + 11{,}59 = 22{,}63\, J/K$$

Näin ollen

$$P = \frac{C \cdot \Delta T}{t} => \Delta T = \frac{P \cdot t}{C}$$

Tässä diplomityössä on käytössä lämpövastus TELPOD HTS–16–230–300 [26], kuva 27.



Kuva 27. *Lämpövastus TELPOD*

Lämpövastus on tehty teräksestä. Sen maksimitoimintajännite on 230V ja teho 300W. Näin ollen voidaan laskea sen resistanssi:

$$I = \frac{U}{R} => R = \frac{U}{I}$$

$$P = U \cdot I => I = \frac{P}{U}$$

Ja lopullinen resistanssin kaava on:

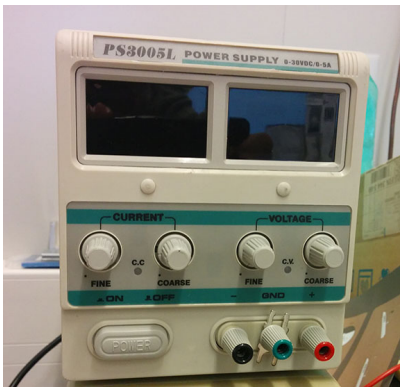
$$R = \frac{U^2}{P}$$

Lasketaan lämpövastuksen resistanssia:

$$R = \frac{U^2}{P} = \frac{230^2}{300} = 176{,}33\Omega$$

Kuitenkin mittaustuloksen mukaan Telpod lämpövastuksen resistanssi on 182 Ω.

Työssä on käytetty virtalähdettä PS3005L, joka on kuvassa 28.



Kuva 28. *Virtalähde*

Virtalähde pystyy tuottamaan maksimissaan 30 V.

Maksimijännite on 30V, ja käytössä olevan lämpövastuksen resistanssi on 182 Ω.

Näin ollen

$$P = U \cdot I = \frac{U^2}{R} = \frac{30^2}{182} = 4{,}95W$$

Palataan takaisin yhtälöön

$$P = 22{,}63\frac{\Delta T}{t}$$

Mutta

$$P = 4{,}95W$$

Tällöin

$$22{,}63\frac{\Delta T}{t} = 4{,}95$$

Jos pitää lämmittää levyä, vaikka minuutin, silloin

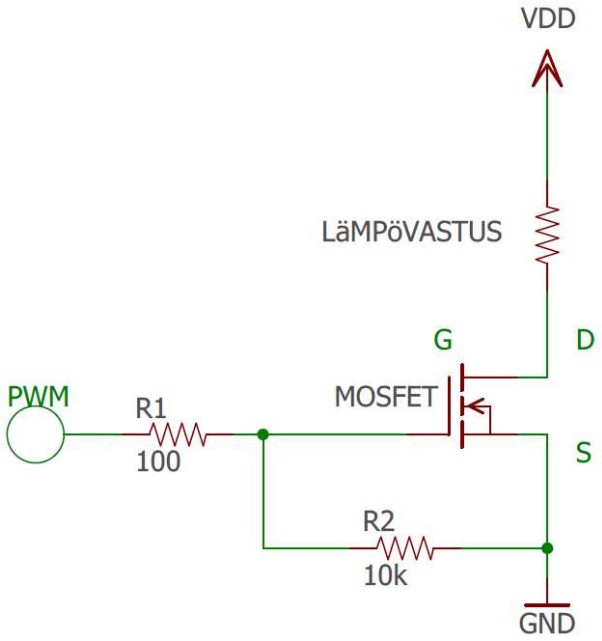
$$\Delta T = \frac{4{,}95 \cdot t}{22{,}63} = \frac{4{,}95 \cdot 60}{22{,}63} = 13{,}12℃$$

Eli levy lämpenee 13,12 °C minuutissa, lämpövastuksen resistanssin ollessa 182 Ω ja jännitteen ollessa 30V.

6.4.7 PWM

Keskusyksikkönä toimiva Arduino-piiri tarjoaa pulssinleveysmodulaatiota (PWM). Sitä voidaan käyttää lämpövastuksen lämmittämiseen. Ongelmana on se, että Arduino pystyy tuottamaan maksimissaan 40mA virtaa. Se ei kuitenkaan riitä komponentin lämmittämiseen.

Ratkaisu siihen ongelmaan on transistorin ohjaus, kuva 29.



Kuva 29. *PWM ja transistori*

Kuormana tässä toimii lämpövastus. PWM mahdollistaa modulointitavan, jossa muuttamalla PWM:n signaalin pulssisuhdetta säädetään kuormaan menevää keskimääräistä tehoa.

R1 vastus suojaa Arduino PWM:n lähdön ja sitä käytetään virtapiikkien välttämiseksi. R2 vastus ns. pull-down vastus auttaa välttämään sellaisia tapauksia, joissa transistori menee päälle ilman ohjausta. FET transistorin ja kytkentäjohtimien resistanssit ovat merkityksettömiä, jolloin kaikki teho kuluu lämmitysvastuksessa.

Tässä työssä PMW ohjaus toimii rajoitetusti ilman säätöä, koska sitä käytetään vain kytkimenä.

6.5 Muut komponentit

Pulssinleveysmodulaatiota varten käytetään yleensä kanavatransistoria. Tässä työssä on käytössä MOSFET NTR4170N.

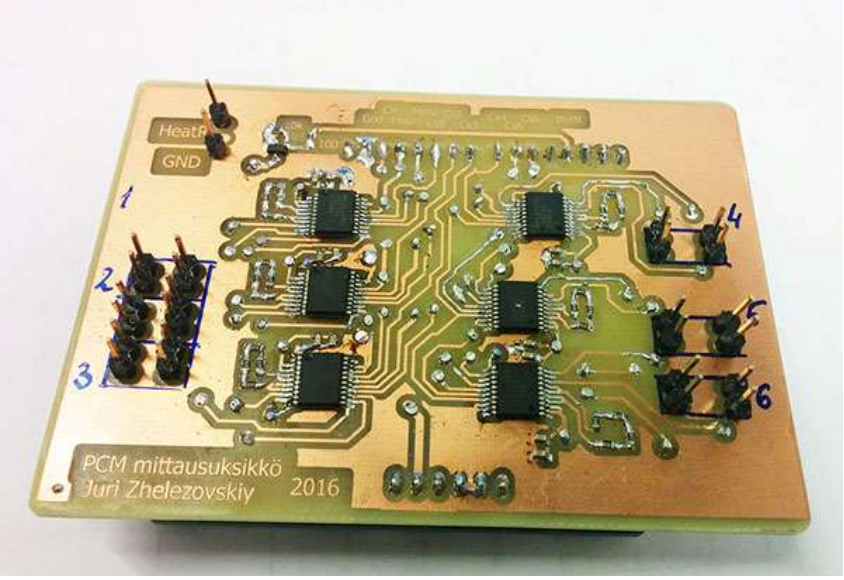
6.6 Fyysinen toteutus

Fyysinen toteutus koskee vain mittausyksikköä, koska keskusyksikköä fyysisesti edustaa Arduino Uno-piiri ja PC-yksikkö on toteutettu kokonaan ohjelmallisesti.

Koko mittausyksikön fyysinen toteutus on esitetty liitteessä A. Kytkentä käyttää nelipistemenetelmää.

6.6.1 Piirilevy

Mittausyksikön valmiin piirilevyn yläpuoli on esitetty kuvassa 30.

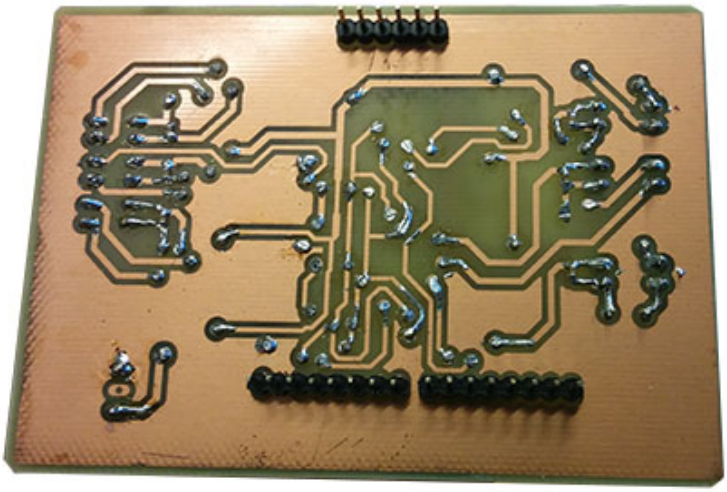


Kuva 30. *Piirilevy komponentteineen (yläpuoli)*

Mittausyksikkö sisältää kuusi MAX mikropiiriä ja muut komponentit. Mittausyksikön lopullinen kytkentäkaavio on esitetty liitteessä B. Kytkentäkaavio on toteutettu Eagle ohjelmistolla. Mittausyksikön layout-suunnitteluun liittyvät kuvat on esitetty liitteissä C ja D.

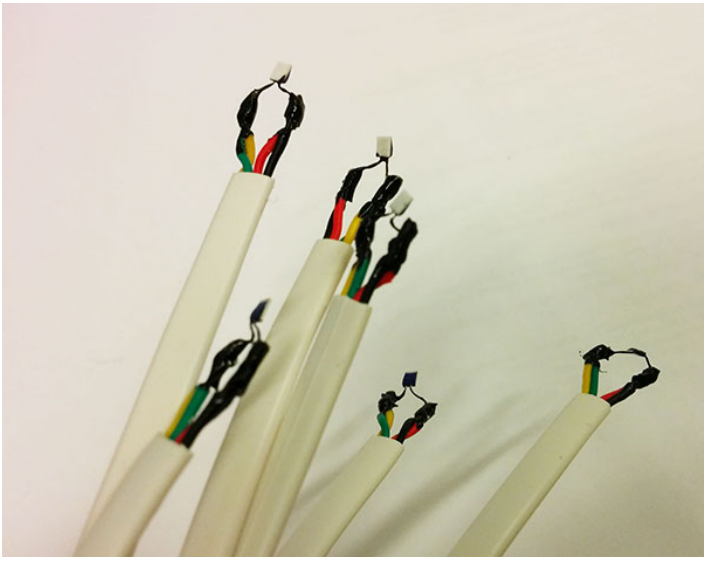
Kuvassa 30 näkyy, että jokaista lämpöanturia kohti on neljä piikkirimaa, koska käytössä on nelipistemenetelmä

Piirilevyn alapuoli juotettujen komponenttien kanssa on esitetty kuvassa 31.



Kuva 31. *Piirilevy komponentteineen (alapuoli)*

6.6.2 Lämpöanturit



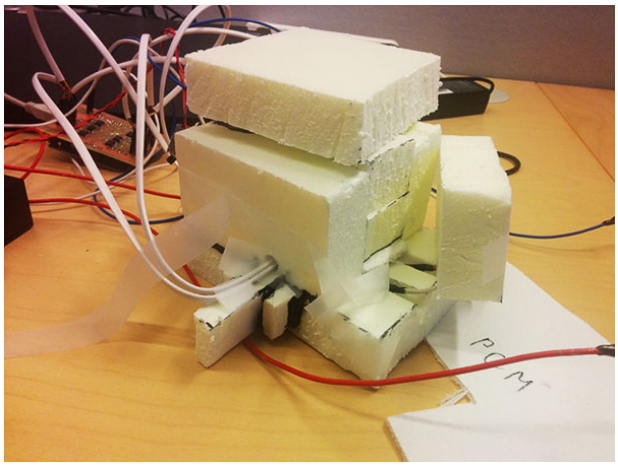
Kuva 32. *Lämpöanturit PT100 kaapeleissa*

Jokaisesta lämpöanturista lähtee neljä johdinta, kuva 32. Juotokset on suojattu muihin komponentteihin oikosuluilta mustalla lakalla, joka on laitettu johtimien päälle.

Jotta lämpöantureiden käsittely helpottuisi, neljä johdinta on laitettu kaapeliin.

6.6.3 Eristeuuni

Kaikki mittaukset suoritetaan uunissa ympäristötekijöiden vaikutuksen välttämiseksi. Eristemateriaaliksi valittiin vaahdotettu polyuretaani. Polyuretaani on polymeeri, ja sopii hyvin tähän tehtävään. Sen lämmönjohtavuus on 0,03 W/(m·K) [30]. Koko uuni on esitetty kuvassa 33.



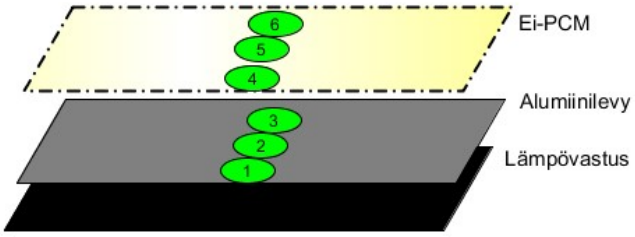
Kuva 33. *Uuni*

7. MITTAUSTULOKSET

Työssä suoritettiin useita mittauksia. Tässä luvussa esitetään muutamat tärkeimmät mittaustulokset.

7.1 Mittaus 1

Mittauksessa 1 tutkittiin alumiinilevyn ja teräslämpövastuksen lämmitys. Kuvassa 34 on esitetty, miten lämpöanturit on aseteltu.

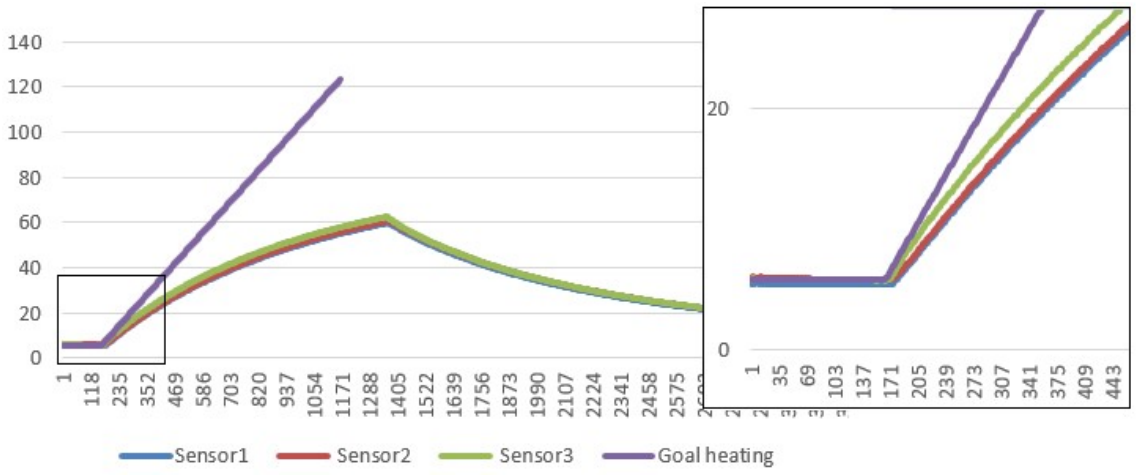


Kuva 34. *Mittauksen 1 skeema*

Lämmitysjännite on 22,1 V ja virta 0,123 A. Lämmitys toimii 2,72 W teholla.

7.1.1 Mittaustulokset

Ensimmäisessä mittauksessa on 3361 näytettä per anturi. Jokainen näyte otetaan kerran sekunnissa. Kuvassa 35 on esitetty lämpöantureiden lämpötila – aika käyrät. Kuvasta on otettu pois neljännen, viidennen ja kuudennen anturin arvot, tuloksen selkeyttämiseksi.



Kuva 35. *Mittaustulokset 1*

Kuvaan on lisätty laskettu lämpötilamuutos (violetti, *Goal heating*), jonka laskeminen on esitetty tulosten pohdinta osassa.

7.1.2 Tulosten pohdinta

Luvussa 6.4.6 laskettiin lämpöero minuutissa jännitteen ollessa 30,0V. Tässä mittauksessa virtalähteestä syöttävä teho on 2,72 W.

Näin ollen lämpötila-nousunopus minuutissa pitää olla

$$\Delta T = \frac{P \cdot t}{C} = \frac{2{,}72\text{W} \cdot 60\text{s}}{22{,}63 Ws/℃} = 7{,}21\ ℃$$

Toivottu lämpötila noudattaa yllä olevan yhtälön tulosta. Kuvasta näkyy se, että alussa alumiinilevyn lämpötila kohoaa laskelman mukaan, mutta sen jälkeen lämpötilan nousu hidastuu. Se johtuu siitä, että uunin eristemateriaali ei eristä täysin aluetta, ja sitoo energiaa. Lisäksi tutkittava materiaali (PCM tai ei-PCM) myös sitoo energiaa. Näin ollen alumiinilevyn energia karkaa.

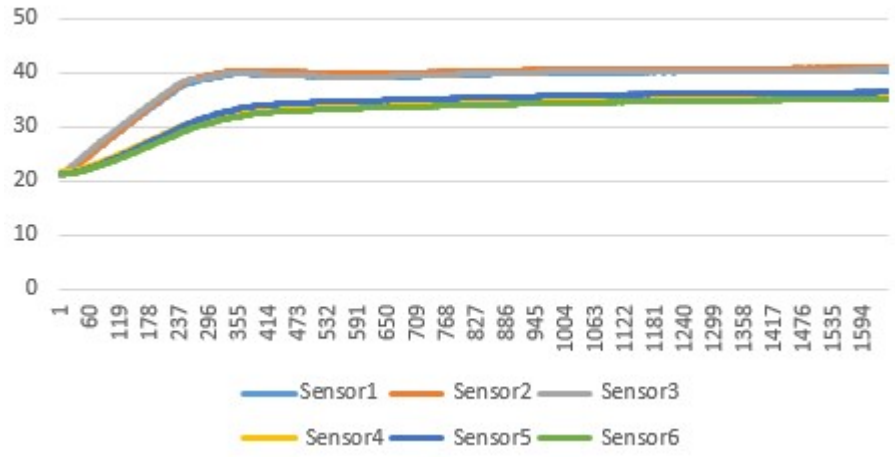
7.1.3 Päätelmä

Mittausten ja laskelmien perusteella voidaan todeta, että lämpökapasiteetti C on laskettu oikein luvussa 6.4.6.

7.2 Mittaus 2

Mittauksessa 2 tutkittiin tarkemmin alumiinilevyn ja teräslämpövastuksen lämmitys. Kuvassa 34 on esitetty, miten lämpöanturit oli aseteltu. Tavoitteena oli tietyllä teholla lämmön tasaantuminen ja pohdinta, kuinka paljon energiaa karkaa tietyllä lämpötilalla.

7.2.1 Mittaustulokset



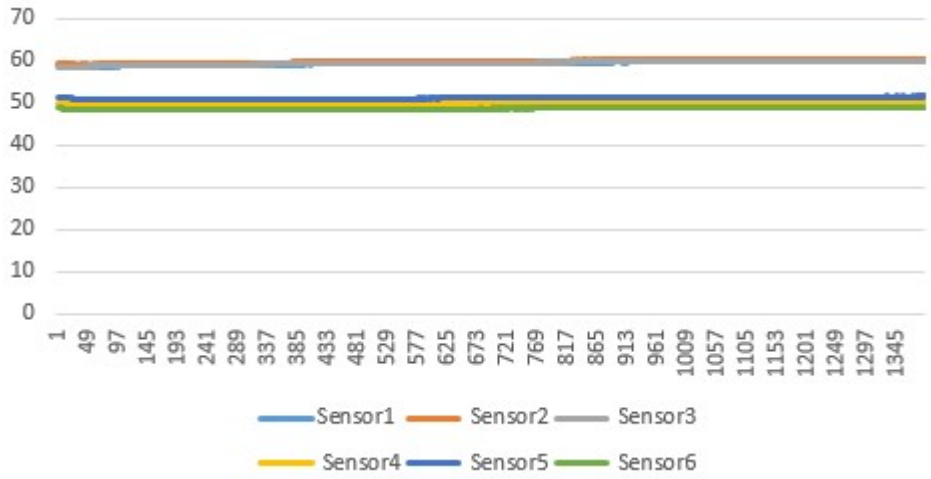
Kuva 36. *Mittaustulos 2, lämpötila 40 ℃*

Ensimmäisessä osassa toivottu lämpötila oli 40 ℃. Sitä varten mittauksessa yritettiin säätää tehoa siten, että toivottu lämpötila pysyisi tasaisena. Kuvassa 36 on esitetty mittaustulokset ja taulukossa 2 jännitteen ja virran arvot, sekä aika ja lämpötila.

Taulukko 2. *Tehon säätäminen, toivottu lämpötila 40℃*

aika, s	1	240	330	600	960
U, V	22,144	15,076	10	11	10,517
I, A	0,122	0,083	0,055	0,061	0,058
T ≈ ,℃	21	38	40	40	40

Tämän jälkeen toivottu lämpötila oli jo 60 ℃. Kuvassa 37 on esitetty tästä osuudesta mittaustulokset.



Kuva 37. *Mittaustulos 2, lämpötila 60 ℃*

Tässä mittauksessa virran ja jännitteen arvot pysyivät vakioina. $I = 0{,}081\ A, U = 14{,}9\ V$

7.2.2 Tulosten pohdinta

Nämä tulokset auttavat ymmärtämään kuinka paljon tehoa karkaa lämpötilan ollessa 40 ℃ ja 60 ℃. Eristyskyky kertoo siitä, kuinka hyvin jokin on eristetty. Mitä parempi eristyskyky on, sitä pienempi on lämmönläpäisykerroin, *k* (kaava 7.2.2.1).

$$k = \frac{P}{\Delta T \cdot A} \tag{7.2.2.1}$$

Pinta-ala *A* on sama ja vakio, näin ollen lasketaan *kA* ensin lämpötilan ollessa 40℃ ja sitten 60℃.

$$kA = \frac{P}{\Delta T}$$

Ympäristölämpötila on sama kahdessa tapauksessa ja se on 21℃.

Lämpötila 40℃: $kA_{40℃} = \frac{10{,}517V \cdot 0{,}058A}{40℃ - 21℃} = 0{,}032$ W/℃

Lämpötila 60℃: $kA_{60℃} = \frac{14{,}9V \cdot 0{,}081A}{60℃ - 2} = 0{,}031$ W/℃

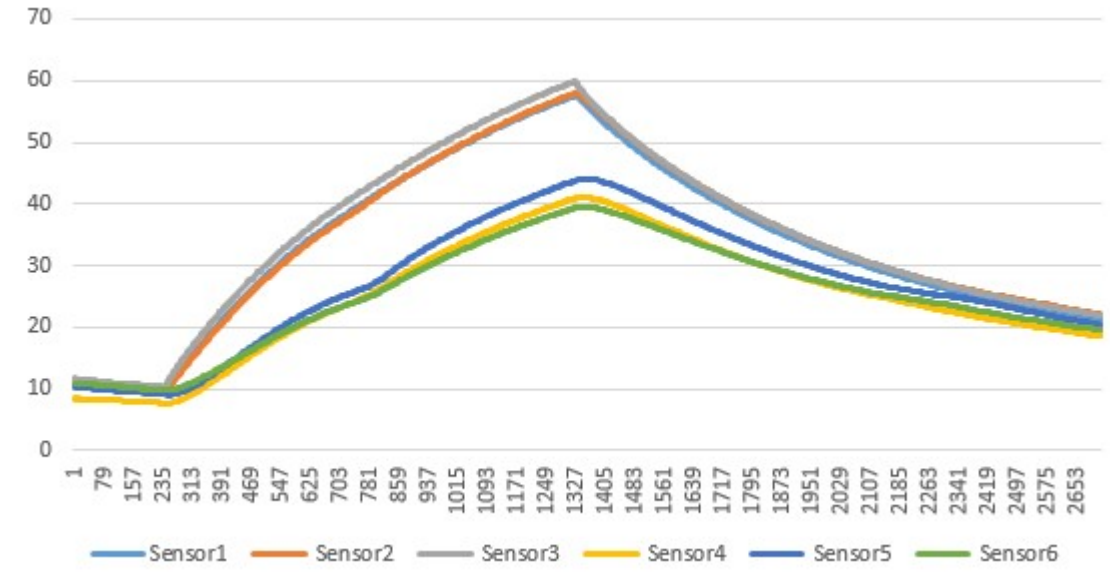
7.2.3 Päätelmä

Uunissa tapahtuva lämpövuoto riippuu lineaarisesti ympäristön ja uunin lämpötilaerosta.

7.3 Mittaus 3

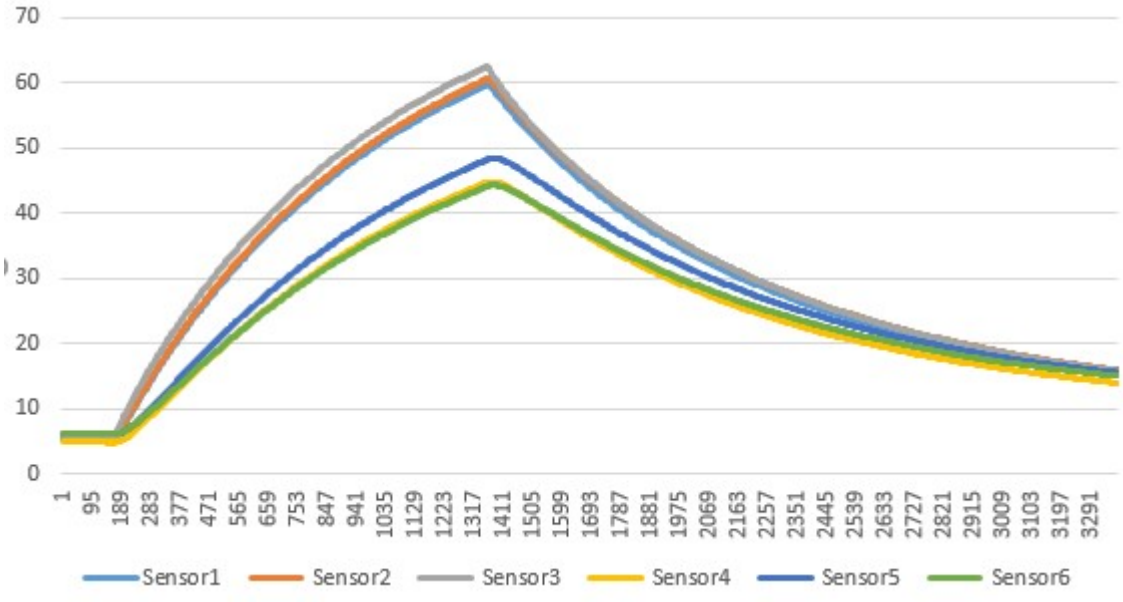
Mittauksessa 3 tutkittiin, kuinka PCM-materiaali on sitonut enemmän energiaa kuin ei-PCM-materiaali.

7.3.1 Mittaustulokset



Kuva 38. *PCM-materiaalin mittaustulokset*

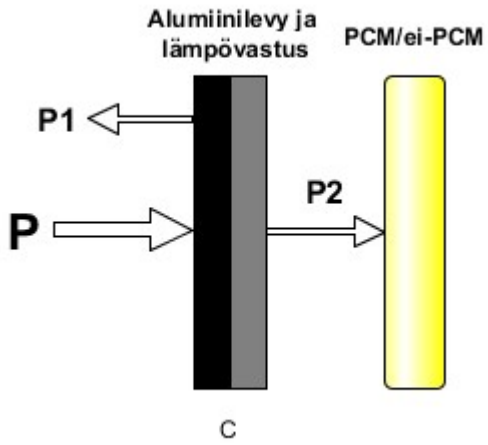
Kuvassa 38 näkyy, miten faasimuutosmateriaali käyttäytyy lämpötilan muuttuessa. Kyseessä on DSC-kalorimetri, joten tarkempaa analyysia varten suoritettiin mittaus, jossa oli ei-PCM-materiaali, kuva 39.



Kuva 39. *ei-PCM-materiaalin mittaustulokset*

7.3.2 Tulosten pohdinta

Aikaisempien mittaustulosten perusteella alettiin analysoida, miksi energia häviää. Tehdään kaksi mittausta, joista ensimmäisessä on PCM-materiaali ja toisessa tavallinen tekstiili.

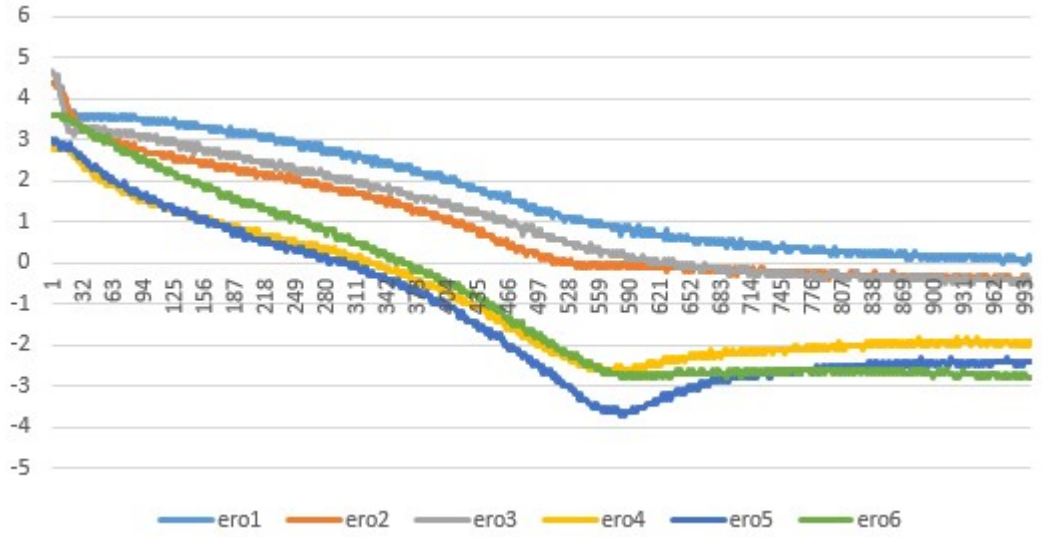


Kuva 40. *Tehon jakautuminen*

Kuvassa 40 näkyy, miten syöttävä teho P jakautuu. Teho P1 johtuu uunin eristemateriaaliin ja vain teho P2 johtuu tutkittavaan materiaaliin.

$$P = P1 + P2$$

Syöttävä teho P ja hukkuva teho P1 ovat samat kahdessa mittauksessa, mutta P2 on eri. P2 tehon suuruus muuttuu, kun kyseessä on PCM-materiaali ja ei-PCM-materiaali. Tämä ominaisuus näkyy kuvissa 38 ja 39. Tätä varten tehtiin kuva 41, jossa on esitetty kahden aikaisemman mittauksen lämpötilaero.



Kuva 41. *Mittausten lämpötilaero*

Kuvassa 41 näkyy se, että PCM-materiaali on sitonut enemmän energiaa kuin ei-PCM. Oletetaan, että kahdessa mittauksessa uuniin karannut energia P1 on sama, koska lämpötila on sama ja syöttävä teho P on sama. Tästä johtuen, voidaan päättää, että PCM-materiaali on sitonut enemmän energiaa.

PCM:

$$\Delta T_{pcm} = \frac{P_{pcm} - P1_{pcm} - P2_{pcm}}{C} \cdot t$$

ei-PCM:

$$\Delta T_{ei-pcm} = \frac{P_{ei-pcm} - P1_{ei-pcm} - P2_{ei-pcm}}{C} \cdot t$$

t on aika ja C on alumiinilevyn ja teräslämpövastuksen yhteinen lämpökapasiteetti, joka laskettiin luvussa 6.4.6.

Todettiin, että $P_{pcm} = P_{ei-pc}$ ja $P1_{pcm} = P1_{ei-}$

Näin ollen:

$$\Delta T_{pcm} - \Delta T_{ei-pc}$$

$$= \frac{P_{pcm}}{C}t - \frac{P1_{pcm}}{C}t - \frac{P2_{pcm}}{C}t - \frac{P_{ei-pcm}}{C}t + \frac{P1_{ei-p}}{C}t + \frac{P2_{ei-p}}{C}t$$

$$= \frac{P2_{ei-pc}}{C}t - \frac{P2_{pcm}}{C}t = \frac{P2_{ei-pc} - P2_{pcm}}{C}t$$

mutta $t \cdot (P2_{ei-pcm} - P2_{pcm}) = \Delta W$, jossa ΔW on energian muutos kahdessa mittauksessa.

Näin ollen:

$$\Delta W = C \cdot (\Delta T_{pcm} - \Delta T_{ei-pcm})$$

Otetaan ensimmäisen kolmen anturin arvot kuvasta 46. Alkuaika on 35 sekuntia ja loppuaika on 1000 sekuntia. Kuvassa 46 näkyy, että lämpötilaero on tasaantunut mittauksen lopussa.

Anturi 1: $\Delta W = 22{,}63 \cdot (3{,}6 - 0{,}1) = 79{,}21\,J$

Anturi 2: $\Delta W = 22{,}63 \cdot (3{,}2 + 0{,}4) = 81{,}47\,J$

Anturi 3: $\Delta W = 22{,}63 \cdot (3{,}2 + 0{,}4) = 81{,}47\,J$

Lasketaan keskiarvo:

$$\Delta W_{keskiarvo} = \frac{79{,}21 + 81{,}47 + 81{,}47}{3} = 80{,}72\,J$$

Seuraavaksi lasketaan PCM-materiaalin sitoma energia grammaa kohti. Punnituksen mukaan PCM-materiaalinäytteen massa on 2,1 gramma ja ei-PCM-materiaalinäytteen 1,5 gramma.

Näin ollen, PCM-materiaalin sitoma lämpömäärä grammaa kohti on:

$$Q_{PCM} = \frac{W}{m} = \frac{80{,}72}{2{,}1} = 38{,}44\,J/g$$

Käytössä on parafiinifaasimuutosmateriaali, jonka tärkein faasimuutosaine on vaha. Lisäksi käytössä oleva ei-PCM-materiaali on rakenteeltaan sama, mutta se ei sisällä vahaa. Näin ollen lasketaan vahan osuus.

$$Q_{PCM\ vaha} = \frac{80{,}72}{(2{,}1 - 1{,}5)} = 134{,}53\,J/g$$

7.3.3 Päätelmä

Mittausten ja laskelmien perusteella voidaan todeta, että PCM-materiaali on sitonut enemmän energiaa (80,72 J) faasimuutoksen aikana kuin ei-PCM.

Lisäksi PCM-materiaali sitoo 38,44 Joulea grammaa kohti, ja faasimuutosmateriaalin sisällä oleva vaha sitoo 134,53 Joulea grammaa kohti.

8. YHTEENVETO

Tässä työssä tutkittiin faasimuutosmateriaaleja ja kalorimetrimittausta. Lisäksi suunniteltiin ja toteutettiin differentiaalinen pyyhkäisykalorimetri.

Työn teoriaosuudessa tutustuttiin lämmönsiirtoon, sen tyyppeihin ja faasimuutoksiin. Lisäksi perehdyttiin faasimuutosmateriaaleihin, niiden toimintaperiaatteeseen, eri tyyppeihin ja sovelluskohteisiin, jotka ovat tekstiilit, kuljettaminen ja rakennukset. Yleisimmät faasimuutosmateriaalien tyypit ovat suolahydraatit ja parafiinit.

Työn etenemisen aikana päätettiin, että DSC-kalorimetri sopii erinomaisesti PCM-materiaalien mittauksiin. Suunnittelu vaiheessa kalorimetri kokonaisuus jaettiin kolmeen moduuliin: PC-yksikköön, keskusyksikköön ja mittausyksikköön. Jokaisella moduulilla on oma roolinsa ja tehtävänsä. Moduuleiden toteutus oli ohjelmallinen ja fyysinen.

DSC-kalorimetrin toteutus onnistui erittäin hyvin ensimmäiseksi versioksi. Kaikki moduulit toteutettiin ns. puhtaalta pöydältä loppuun asti.

Kalorimetri mahdollistaa faasimuutosmateriaalien mittaukset ja tutkimukset. Tässä työssä mittausten avulla tutkittiin käytössä olevaa PCM-materiaalia.

Työssä mitattiin myös lämpötiloja näytteen toiselta puolelta. Näiden tulosten perusteella on mahdollista tutkia jatkossa myös materiaalien lämmöneristävyyttä.

Toiselta puolelta mitattuna käyrissä näkyy huomattavasti suurempi muutos. Käytännössä, jos tätä kangasmateriaalia käytetään vaatteessa, toinen puoli on ihoa vasten, jolloin muutokset tuntuvat selkeämmin. PCM-materiaalin kanssa lämpötila saattaa mittauksen perusteella olla hetkellisesti jopa 4-5 astetta alhaisempi.

LÄHTEET

[1] Arduino, analogRead [WWW].[viitattu 14.11.2016]. Saatavissa: https://www.arduino.cc/en/Reference/AnalogRead

[2] Arduino, Main page [WWW].[viitattu 8.11.2016]. Saatavissa: https://www.arduino.cc/

[3] Arduino, Serial [WWW].[viitattu 9.11.2016]. Saatavissa: https://www.arduino.cc/en/Reference/Serial

[4] Arduino Tutorials [WWW].[viitattu 03.09.2016]. Saatavissa: https://www.arduino.cc/en/Tutorial/HomePage

[5] Atkins luku 2, Termodynmiikan 1. pääsääntö, Jyväskylän yliopisto [WWW] [viitattu 8.12.2016]. Saatavissa: http://users.jyu.fi/~hahakkin/opetus/KEMA221-2009/Atkins-luku2.pdf

[6] Atmel, Atmega328 data sheet. Saatavissa: http://www.atmel.com/Images/Atmel-8271-8-bit-AVR-Microcontroller-ATmega48A-48PA-88A-88PA-168A-168PA-328-328P_datasheet_Complete.pdf, Viitattu 9.11.2016

[7] BBC – GCSE Bitesize: Measuring Energy Transfers [WWW] [viitattu 3.11.2016]. Saatavissa: http://www.bbc.co.uk/schools/gcsebitesize/science/ocr_gateway_pre_2011/carbon_chem/8_energy2.shtml

[8] Y. A. Cengel, Heat transfer: a practical approach, Tom Casson, R. R. Donnelley & Sons Company, 1998.

[9] Y. A. Cengel, Introduction to thermodynamics and heat transfer, The McGraw-Hill Company, 1997

[10] ChemEd Dl, Chemical Education Digital Library, Enthalpy of Fusion and Enthalpy of Vaporization [WWW] [viitattu 24.10.2016]. Saatavissa: http://chempaths.chemeddl.org/services/chempaths/?q=book/General%20Chemistry%20Textbook/Solids%2C%20Liquids%20and%20Solutions/1406/enthalpy-fusion-and-enthalpy-vapo

[11] T. Engel, Thermodynamics, statistical thermodynamics, and kinetics, 3. Painos, Pearson, 2006.

[12] T. Hasenöhrl, An introduction to Phase Change Materials as Heat Storage Mediums, 2009, Saataviss: http://www.lth.se/fileadmin/ht/Kurser/MVK160/2009_reports/ThomasHasenohrl_PCMs.pdf, Viitattu18.08.2016.

[13] Heraeus, M222, Platinum Resistance temperature detector, data sheet. Saatavissa: http://www.mouser.com/ds/2/619/M222_HST-USA-270818.pdf, Viitattu: 14.11.2016.

[14] G. W. H. Höhne, W. F. Hemminger, H.-J. Flammersheim, Differential scanning calorimetry, 2. Painos, Springer, 2003.

[15] M. Inkinen, Mikrokontrollerit, Kurssi ELT-21300, luentokalvot, Tampereen teknillinen yliopisto, 2015.

[16] Keithley, A Tektronix Company, Two-Wire vs. Four-Wire Resistance Measurements: Which Configuration Makes Sense for Your Application? [WWW] [viitattu 15.11.2016]. Saatavissa: http://www.tek.com/sites/tek.com/files/media/document/resources/2Wire_4Wire%20Resistance%20Article.pdf

[17] Labfacility, DM-317 Thermocouple data sheet. Saatavissa: http://www.farnell.com/datasheets/1918818.pdf?_ga=1.258207310.596730693.1471508194, Viitattu: 23.08.2016.

[18] Livescience, What Are Calories? [WWW] [viitattu 24.10.2016]. Saatavissa: http://www.livescience.com/52802-what-is-a-calorie.html

[19] LUT, Lppeenranta University of technology, BM30A0240, Fysiikka L osa 2 [WWW] [viitattu 24.10.2016]. Saatavissa: http://www.mafy.lut.fi/study/HeikkiP/Fysiikka-L-osa-2/luento1.pdf

[20] Maxim integrated products, MAX31865 data sheet. Saatavissa: http://www.farnell.com/datasheets/2002033.pdf?_ga=1.216640082.596730693.1471508194, Viitattu: 29.08.2016.

[21] New World Encyclopedia, Calorimeter [WWW] [viitattu 3.11.2016]. Saatavissa: http://www.newworldencyclopedia.org/entry/Calorimeter

[22] Noppa, Oulun yliopisto, Kalorimetri [WWW] [viitattu 1.11.2016]. Saatavissa: https://noppa.oulu.fi/noppaimages/766106P/KALORIMETRI.pdf

[23] Pharmaceutical Outsourcing, Phase Change Materials, A Brief Comparison of Ice Packs, Salts, Paraffins, and Vegetable-derived Phase Change Materials, William R. Sutterlin, Ph.D, 2011 [WWW] [viitattu 25.10.2016]. Saatavissa: http://www.pharmoutsourcing.com/Featured-Articles/37854-Phase-Change-Materials-A-Brief-Comparison-of-Ice-Packs-Salts-Paraffins-and-Vegetable-derived-Phase-Change-Materials/

[24] Pielichowska, K; Pielichowski, K, Phase change materials for thermal energy storage, 2014, progress in materials science, Saatavissa: http://dx.doi.org/10.1016/j.pmatsci.2014.03.005

[25] Polymer Science Learning Center, Differential Scanning Calorimetry [WWW] [viitattu 3.11.2016]. Saatavissa: http://www.pslc.ws/macrog/dsc.htm

[26] Telpod, HTS-16-230-300 ominaisuudet [WWW] [viitatttu 15.09.2016]. Saatavissa: http://www.tme.eu/fi/details/hts16-230-300/lampovastukset/telpod/hts-16-230-300-363/

[27] The Engineering ToolBox, Densities of Solids [WWW] [viitattu 15.11.2016]. Saatavissa: http://www.engineeringtoolbox.com/density-solids-d_1265.html

[28] The Engineering ToolBox, Latent Heat of Melting of some common Materials [WWW] [viitattu 25.10.2016]. Saatavissa: http://www.engineeringtoolbox.com/latent-heat-melting-solids-d_96.html

[29] The Engineering ToolBox, Specific Heat of common Substances [WWW] [viitattu 24.10.2016]. Saatavissa: http://www.engineeringtoolbox.com/specific-heat-capacity-d_391.html

[30] The Engineering ToolBox, Thermal Conductivity of some common Materials and Gases [WWW] [viitattu 06.10.2016]. Saatavissa: http://www.engineeringtoolbox.com/thermal-conductivity-d_429.html

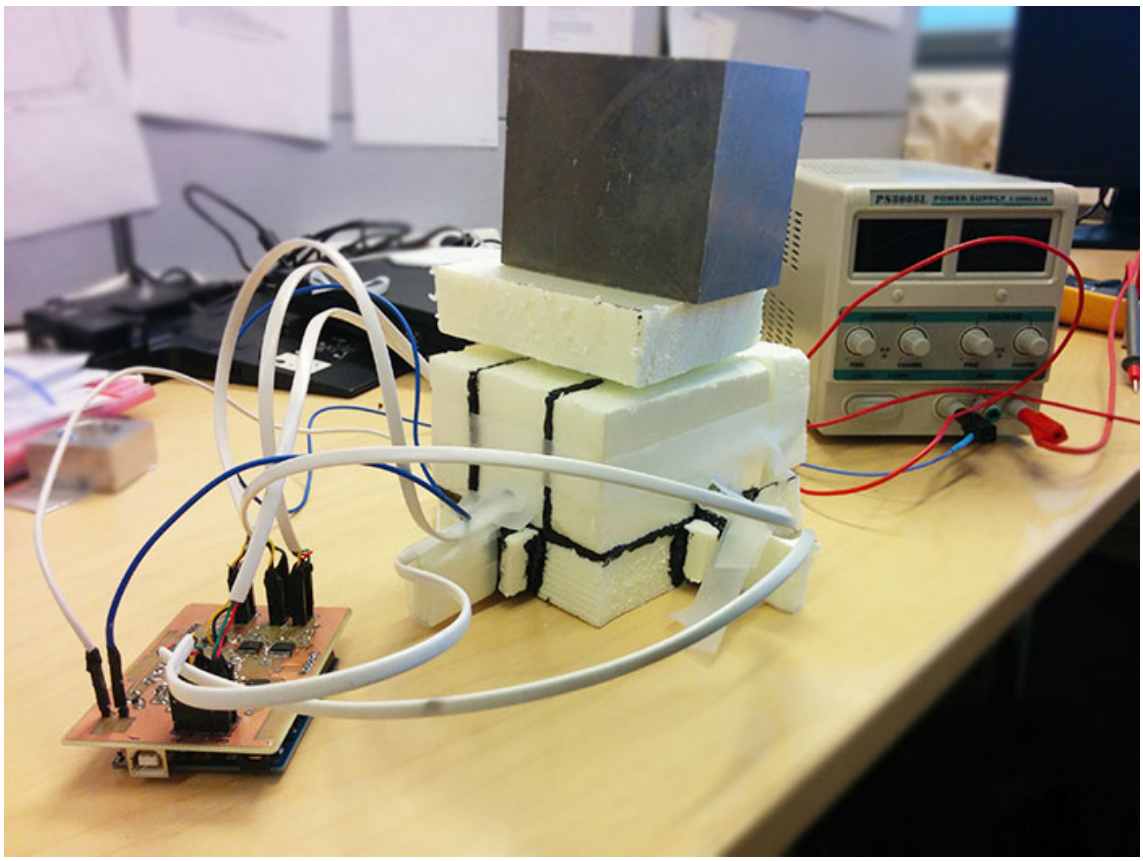
[31] The Engineering ToolBox, Water Steam - the Critical Point [WWW] [viitattu 21.10.2016]. Saatavissa: http://www.engineeringtoolbox.com/critical-point-water-steam-d_834.html

[32] W. Wagner, suomentanut O. Ranta, Wärmeübertragung/Lämmönsiirto, Opetushallitus, Painatuskeskus Oy, Helsinki, 1994.

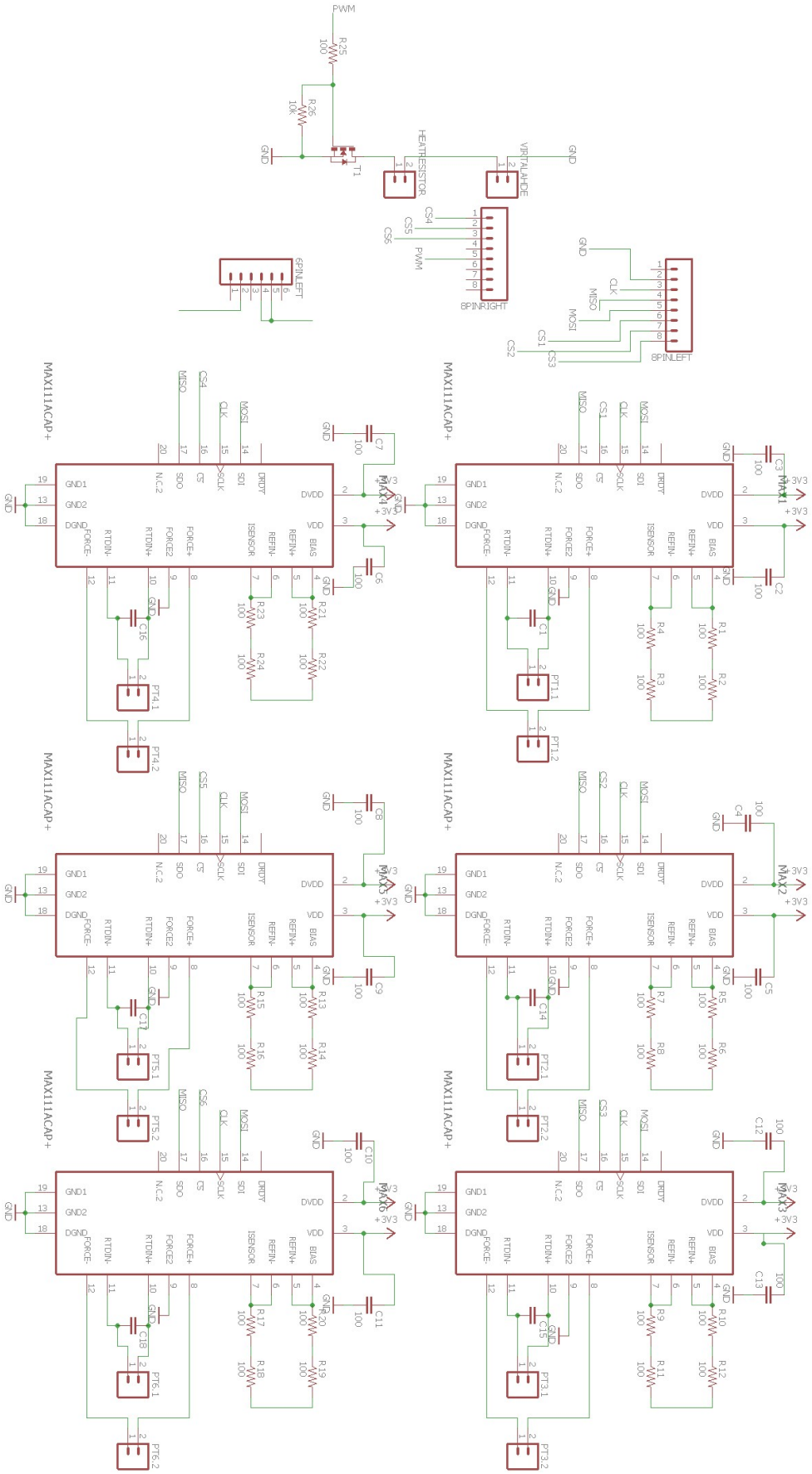
[33] Velkuan koulu, Naantali, Ominaislämpökapasiteetti [WWW] [viitattu 24.10.2016]. Saatavissa: https://peda.net/naantali/velkuan-koulu/oppiaineet2/fysiikka/efysiikka-83f/1loarl/o

[34] Теплопередача. Виды теплопередачи. Теплопроводность. Heat transfer. Types of heat transfer. Thermal conductivity. [WWW] [viitattu 8.12.2016]. Saatavissa: http://class-fizika.narod.ru/8_3.htm

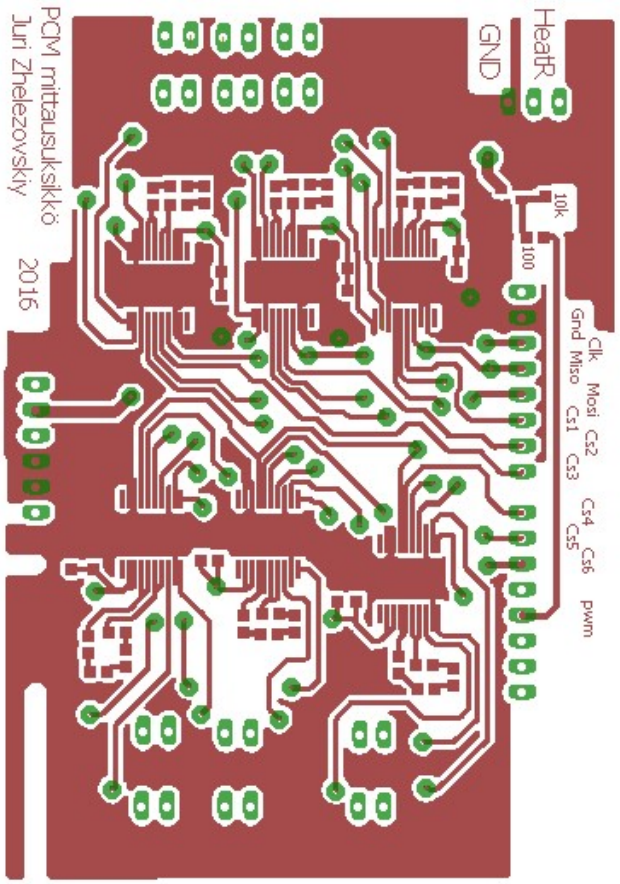
LIITE A: MITTAUSYKSIKÖN TOTEUTUS



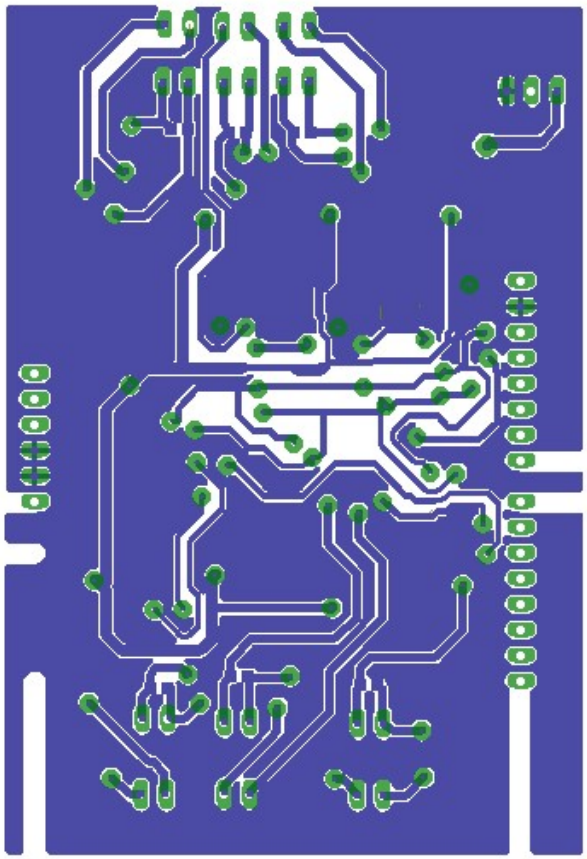
LIITE B: MITTAUSYKSIKÖN KYTKENTÄKAAVIO



LIITE C: MITTAUSYKSIKÖN LAYOUT TOP



LIITE D: MITTAUSYKSIKÖN LAYOUT BOTTOM



LIITE E: KESKUSYKSIKÖN OHJELMA

```
/*********************************************************************
 * Aduino kommunikoi sarjaliikenteen kautta PC:n kanssa.
 * Saadaan dataa PT100 antureista (6 kpl) MAXIM31865 piirien kautta
 * Get information from PT100 sensors by using MAX 31865
 * Set up heating element
 * Versio: 3.2
 * 2016
 * Juri Zhelezovskiy
*********************************************************************/

#include <SPI.h>
#include <stdint.h>
#include <Arduino.h>

// find possible faults in measurements, datasheet table 7, p.16
#define MAX31865_IS_RTD_HIGH_THRESHOLD_FAULT  ( 1 << 7 ) // 0b 1000
0000
#define MAX31865_IS_RTD_LOW_THRESHOLD_FAULT   ( 1 << 6 ) // 0b 0100
0000
#define MAX31865_IS_REFIN_MORE_085_FAULT      ( 1 << 5 ) // 0b 0010
0000
#define MAX31865_IS_REFING_LESS_085_FAULT     ( 1 << 4 ) // 0b 0001
0000
#define MAX31865_IS_RTDIN_LESS_085_FAULT      ( 1 << 3 ) // 0b 0000
1000
#define MAX31865_IS_VOLTAGE_FAULT             ( 1 << 2 ) // 0b 0000
0100

// IEC 751 määrittää Callendar-Van Dusen kaavan arvoja näin, constants
for formula
#define RTD_A         (3.90830e-3)
#define RTD_B         (-5.77500e-7)

// Reference resistor
#define RTD_RREF_PT100     (400)

// Chip Select pins for MAX31865
#define CS_PIN_1 (10)
#define CS_PIN_2 (9)
#define CS_PIN_3 (8)
#define CS_PIN_4 (7)
#define CS_PIN_5 (6)
#define CS_PIN_6 (5)

// pin for PWM to transistor
#define FET_PIN (3)

/**
 * luokka, jonka jokainen olio edustaa yhtä MAX piiriä
 * class, where each object is a MAX31865
**/

class MAX31865_piiri
{
public:
  MAX31865_piiri( uint8_t cs_pin );
  void asetukset( void );
  uint8_t lue_kaikki( void );
```

```
  double lampotilanlasku( ) const;

  uint8_t status( )           const { return( mitattu_status ); }
  uint16_t low_threshold( )   const { return( mitattu_low_threshold );
}
  uint16_t high_threshold( )  const { return( mitattu_high_threshold
); }
  uint16_t raw_resistance( )  const { return( mitattu_resistance ); }

  double resistance( )        const
  {
    // mitattu_resistanssi * 400 /32768, ADC resoluutio, 2^15 = 32 768
    // datasheet p.19, R(RTD) = ( ADC code x Rref ) / 2^15
    return( (double)raw_resistance() * RTD_RREF_PT100 / 32768 );
  }

private:
  // MAXin CS pinni, jota käytetään tässä luokassa
  uint8_t  cs_pin;

  // data MAXimista
  uint8_t  mitattu_configuration;
  uint16_t mitattu_resistance;
  uint16_t mitattu_high_threshold;
  uint16_t mitattu_low_threshold;
  uint8_t  mitattu_status;
};

// create object of class
MAX31865_piiri::MAX31865_piiri( uint8_t cs_pin )
{
  // otetaan talteen CS pinni
  this->cs_pin = cs_pin;
  pinMode( this->cs_pin, OUTPUT );
  // ensin se on ylhäällä
  digitalWrite( this->cs_pin, HIGH );
}

// määritetään asetukset MAXille
// set up MAX31865
void MAX31865_piiri::asetukset( void )
{
    uint8_t asetus_bitit = 0;
    /**
     * 0xC3 = 0b 1100 0011
     * D7 = 1: Vbias 1:on 0:off
     * D6 = 1: Conversion mode 1:Auto 0:off
     * D5 = 0: 1-shot 1:on 0:off
     * D4 = 0: 1:3 Wire, 0:2/4 Wire
     * D3, D2: Fault detection
     * D1 = 1: Fault status 1:true
     * D0 = 1: 1:50Hz, 0:60Hz
    */
    asetus_bitit = 0xC3;


    // send settings to MAX
    digitalWrite( this->cs_pin, LOW );
    // lähetetään konfigurointi asetukset, koska se on write to MAX,
osoite on 0x8h, sivu 13
    SPI.transfer( 0x80 );
```

```
    SPI.transfer( asetus_bitit );
    digitalWrite( this->cs_pin, HIGH );
}
/*
 * Lasketaan lämpötila
 * Count temperature by using formula
 * Callendar-Van Dusen kaavan mukaan resistanssi on
 * R(T) = R0( 1 + aT + bT^2 + c(T - 100)*T^3 )
 * mutta
 * jos lämpötila on plussan puolella -> c = 0, tälloin
 * R(T) = R0( 1 + aT  + bT^2 ), siitä tulee
 * T = (   -a + sqrt(  a*a - 4b( 1 - R(t)/R(0) )  )   )/2b
 */
double MAX31865_piiri::lampotilanlasku( ) const
{
  // 2*B
  static const double b2   = 2.0 * RTD_B;
  // A^2
  static const double aa = RTD_A * RTD_A;

  // 1 - R(t)/R(0), R(t) tällä hetkellä, R(0) = 100, koska se on PT100
  double c = 1.0 - resistance() / 100;
  double sqrt_arvo = sqrt ( aa - 2.0 * b2 * c );
  double aste = ( -RTD_A + sqrt_arvo ) / b2;

 // temperature in Celcius
  return( aste );
}

/**
 * luetaan rekistereita jokainen kerrallaan (table 1 sivu 13 )
 * Sitten otetaan ne talteen.
 * Palautetaan stutus bitit.
 * Read from registers and store them. Return status bytes
 */
uint8_t MAX31865_piiri::lue_kaikki( void )
{
  uint16_t rtd_bitit;

  // CS alhaalla ja kommunikointi alkaa
  digitalWrite( this->cs_pin, LOW );
  // we want to read from 0 register
  // lähetetään MAXiin 0x00, joka tarkoittaa, että 1. haluamme lukea,
koska se on lähetys ja 2. haluamme lukea bitistä 0 alkaen, sivu 17
  SPI.transfer( 0x00 );

  /* Read the MAX31865 registers
   *  1. Configuration
   *  2. RTD
   *  3. High Fault Threshold
   *  4. Low Fault Threshold
   *  5. Fault Status
   */
  // 1. configuration register, 8 bittiä, 0:sta alkaen
  this->mitattu_configuration = SPI.transfer( 0x00 );

  // 2. RTD resistance registers MSBs(8 bittiä) ja LSBs(8 bittiä), yh-
teensä 16 bittiä
  // luetaan seuraavat 8 bittiä ja shiftataan vasemmalle
  rtd_bitit  = SPI.transfer( 0x00 ) << 8;
  // luetaan 8 bittiä ja lisätään oikealle
```

```
  rtd_bitit |= SPI.transfer( 0x00 );
  // tallennetaan data, paitsi 1 bitti, jota emme tarvitse. sivu 15
  this->mitattu_resistance = rtd_bitit >> 1;

  // 3. luetaan High Fault Threshold, 16 bittiä
  rtd_bitit  = SPI.transfer( 0x00 ) << 8;
  rtd_bitit |= SPI.transfer( 0x00 );
  this->mitattu_high_threshold = rtd_bitit ;

  // 4. luetaan Low Fault Threshold, 16 bittiä
  rtd_bitit  = SPI.transfer( 0x00 ) << 8;
  rtd_bitit |= SPI.transfer( 0x00 );
  this->mitattu_low_threshold = rtd_bitit ;

  // 5. luetaan fault status, vain 8 bittiä
  this->mitattu_status = SPI.transfer( 0x00 );

  // CS yllhaalla ja kommunikointi loppuu
  digitalWrite( this->cs_pin, HIGH );

  // Resetoidaan asetukset jos tulee virheitä lukemisessa
  if(   this->mitattu_resistance == 0 || this->mitattu_status != 0  )
    asetukset( );

  return( status() );
}

// luodaan luokan olioita
// create objects of class
MAX31865_piiri sensor1( MAX31865_piiri( CS_PIN_1 ) );
MAX31865_piiri sensor2( MAX31865_piiri( CS_PIN_2 ) );
MAX31865_piiri sensor3( MAX31865_piiri( CS_PIN_3 ) );
MAX31865_piiri sensor4( MAX31865_piiri( CS_PIN_4 ) );
MAX31865_piiri sensor5( MAX31865_piiri( CS_PIN_5 ) );
MAX31865_piiri sensor6( MAX31865_piiri( CS_PIN_6 ) );

void setup()
{
  Serial.begin( 115200 );

  // SPI asetukset
  SPI.begin( );
  SPI.setClockDivider( SPI_CLOCK_DIV16 );
  // MAX31865 käyttää SPI 1 ja SPI 3, sivu 16 datalehti
  SPI.setDataMode( SPI_MODE3 );


  sensor1.asetukset(  );
  sensor2.asetukset(  );
  sensor3.asetukset(  );
  sensor4.asetukset(  );
  sensor5.asetukset(  );
  sensor6.asetukset(  );
  // give the sensor time to set up
  delay(100);

  pinMode( FET_PIN, OUTPUT );
}

double temperature = 0;
int nro = 1;
```

```
void loop()
{
  while ( Serial.available() == 0 );
  char letter = Serial.read();
  // wait for symbol to start
  if( letter == '1' )
  {
    Serial.println("Mittaustulokset:");
    while( true )
    {
      sensor1.lue_kaikki( );
      sensor2.lue_kaikki( );
      sensor3.lue_kaikki( );
      sensor4.lue_kaikki( );
      sensor5.lue_kaikki( );
      sensor6.lue_kaikki( );

      // switch PWM on
      digitalWrite( FET_PIN, 255 );

      Serial.print( "Mittaus nro, measure " );
      Serial.println( nro );
      nro++;
       /*Sensor PT100 Nro 1*/
      // jos fault status bitit ovat nollia, eli ei ole virheitä
      if( sensor1.status( ) == 0 )
      {
        // get temperature
        temperature = sensor1.lampotilanlasku( );
        // print results
        Serial.print( " T1 = "); Serial.print( temperature ); Se-
rial.println(" C" );
      }
      // on virheitä, faults
      else
      {
        Serial.print( "S.1 Verheita, faults: " );
        Serial.print( sensor1.status( ) );
        Serial.print( ": " );
        // Datasheet table 7
        // find out faults
        if( sensor1.status( ) & MAX31865_IS_RTD_HIGH_THRESHOLD_FAULT )
        {
          Serial.println( "Measured resistance of PT100 greater than
HIGH Fault Threshold value" );
        }
        else if( sensor1.status( ) & MAX31865_IS_RTD_LOW_THRESH-
OLD_FAULT )
        {
          Serial.println( "Measured resistance of PT100 less than LOW
Fault Threshold value" );
        }
        else if( sensor1.status( ) & MAX31865_IS_REFIN_MORE_085_FAULT
)
        {
          Serial.println( "REF(IN-) > 0.85 x V(BIAS)" );
        }
        else if( sensor1.status( ) & MAX31865_IS_REFING_LESS_085_FAULT
)
        {
```

```
          Serial.println( "REF(IN-) < 0.85 x V(BIAS), FORCE(-) is
open" );
        }
        else if( sensor1.status( ) & MAX31865_IS_RTDIN_LESS_085_FAULT
)
        {
          Serial.println( "RTD(IN-) < 0.85 x V(BIAS), FORCE(-) is
open" );
        }
        else if( sensor1.status( ) & MAX31865_IS_VOLTAGE_FAULT )
        {
          Serial.println( "Input voltage > V(DD) or < GND");
        }
        else
        {
          Serial.println( "Tuntematon virhe/unknown fault" );
        }
      }

      /*Sensor PT100 Nro 2*/
      // jos fault status bitit ovat nollia, eli ei ole virheitä
      if( sensor2.status( ) == 0 )
      {
        // get temperature
        temperature = sensor2.lampotilanlasku( );
        // print results
        Serial.print( " T2 = "); Serial.print( temperature ); Se-
rial.println(" C" );
      }
      // on virheitä, faults
      else
      {
        Serial.print( "S2. Verheita, faults: " );
        Serial.print( sensor2.status( ) );
        Serial.print( ": " );
        // Datasheet table 7
        // find out faults
        if( sensor2.status( ) & MAX31865_IS_RTD_HIGH_THRESHOLD_FAULT )
        {
          Serial.println( "Measured resistance of PT100 greater than
HIGH Fault Threshold value" );
        }
        else if( sensor2.status( ) & MAX31865_IS_RTD_LOW_THRESH-
OLD_FAULT )
        {
          Serial.println( "Measured resistance of PT100 less than LOW
Fault Threshold value" );
        }
        else if( sensor2.status( ) & MAX31865_IS_REFIN_MORE_085_FAULT
)
        {
          Serial.println( "REF(IN-) > 0.85 x V(BIAS)" );
        }
        else if( sensor2.status( ) & MAX31865_IS_REFING_LESS_085_FAULT
)
        {
          Serial.println( "REF(IN-) < 0.85 x V(BIAS), FORCE(-) is
open" );
        }
        else if( sensor2.status( ) & MAX31865_IS_RTDIN_LESS_085_FAULT
)
```

```
        {
          Serial.println( "RTD(IN-) < 0.85 x V(BIAS), FORCE(-) is
open" );
        }
        else if( sensor2.status( ) & MAX31865_IS_VOLTAGE_FAULT )
        {
          Serial.println( "Input voltage > V(DD) or < GND");
        }
        else
        {
          Serial.println( "Tuntematon virhe/unknown fault" );
        }
      }


      /*Sensor PT100 Nro 3*/
      // jos fault status bitit ovat nollia, eli ei ole virheitä
      if( sensor3.status( ) == 0 )
      {
        // get temperature
        temperature = sensor3.lampotilanlasku( );
        // print results
        Serial.print( " T3 = "); Serial.print( temperature ); Se-
rial.println(" C" );
      }
      // on virheitä, faults
      else
      {
        Serial.print( "S3. Verheita, faults: " );
        Serial.print( sensor3.status( ) );
        Serial.print( ": " );
        // Datasheet table 7
        // find out faults
        if( sensor3.status( ) & MAX31865_IS_RTD_HIGH_THRESHOLD_FAULT )
        {
          Serial.println( "Measured resistance of PT100 greater than
HIGH Fault Threshold value" );
        }
        else if( sensor3.status( ) & MAX31865_IS_RTD_LOW_THRESH-
OLD_FAULT )
        {
          Serial.println( "Measured resistance of PT100 less than LOW
Fault Threshold value" );
        }
        else if( sensor3.status( ) & MAX31865_IS_REFIN_MORE_085_FAULT
)
        {
          Serial.println( "REF(IN-) > 0.85 x V(BIAS)" );
        }
        else if( sensor3.status( ) & MAX31865_IS_REFING_LESS_085_FAULT
)
        {
          Serial.println( "REF(IN-) < 0.85 x V(BIAS), FORCE(-) is
open" );
        }
        else if( sensor3.status( ) & MAX31865_IS_RTDIN_LESS_085_FAULT
)
        {
          Serial.println( "RTD(IN-) < 0.85 x V(BIAS), FORCE(-) is
open" );
        }
```

```
        else if( sensor3.status( ) & MAX31865_IS_VOLTAGE_FAULT )
        {
          Serial.println( "Input voltage > V(DD) or < GND");
        }
        else
        {
          Serial.println( "Tuntematon virhe/unknown fault" );
        }
      }

      /*Sensor PT100 Nro 4*/
      // jos fault status bitit ovat nollia, eli ei ole virheitä
      if( sensor4.status( ) == 0 )
      {
        // get temperature
        temperature = sensor4.lampotilanlasku( );
        // print results
        Serial.print( " T4 = "); Serial.print( temperature ); Se-
rial.println(" C" );
      }
      // on virheitä, faults
      else
      {
        Serial.print( "S4. Verheita, faults: " );
        Serial.print( sensor4.status( ) );
        Serial.print( ": " );
        // Datasheet table 7
        // find out faults
        if( sensor4.status( ) & MAX31865_IS_RTD_HIGH_THRESHOLD_FAULT )
        {
          Serial.println( "Measured resistance of PT100 greater than
HIGH Fault Threshold value" );
        }
        else if( sensor4.status( ) & MAX31865_IS_RTD_LOW_THRESH-
OLD_FAULT )
        {
          Serial.println( "Measured resistance of PT100 less than LOW
Fault Threshold value" );
        }
        else if( sensor4.status( ) & MAX31865_IS_REFIN_MORE_085_FAULT
)
        {
          Serial.println( "REF(IN-) > 0.85 x V(BIAS)" );
        }
        else if( sensor4.status( ) & MAX31865_IS_REFING_LESS_085_FAULT
)
        {
          Serial.println( "REF(IN-) < 0.85 x V(BIAS), FORCE(-) is
open" );
        }
        else if( sensor4.status( ) & MAX31865_IS_RTDIN_LESS_085_FAULT
)
        {
          Serial.println( "RTD(IN-) < 0.85 x V(BIAS), FORCE(-) is
open" );
        }
        else if( sensor4.status( ) & MAX31865_IS_VOLTAGE_FAULT )
        {
          Serial.println( "Input voltage > V(DD) or < GND");
        }
        else
```

```
        {
          Serial.println( "Tuntematon virhe/unknown fault" );
        }
      }
      /*Sensor PT100 Nro 5*/
      // jos fault status bitit ovat nollia, eli ei ole virheitä
      if( sensor5.status( ) == 0 )
      {
        // get temperature
        temperature = sensor5.lampotilanlasku( );
        // print results
        Serial.print( " T5 = "); Serial.print( temperature ); Se-
rial.println(" C" );
      }
      // on virheitä, faults
      else
      {
        Serial.print( "S5. Verheita, faults: " );
        Serial.print( sensor5.status( ) );
        Serial.print( ": " );
        // Datasheet table 7
        // find out faults
        if( sensor5.status( ) & MAX31865_IS_RTD_HIGH_THRESHOLD_FAULT )
        {
          Serial.println( "Measured resistance of PT100 greater than
HIGH Fault Threshold value" );
        }
        else if( sensor5.status( ) & MAX31865_IS_RTD_LOW_THRESH-
OLD_FAULT )
        {
          Serial.println( "Measured resistance of PT100 less than LOW
Fault Threshold value" );
        }
        else if( sensor5.status( ) & MAX31865_IS_REFIN_MORE_085_FAULT
)
        {
          Serial.println( "REF(IN-) > 0.85 x V(BIAS)" );
        }
        else if( sensor5.status( ) & MAX31865_IS_REFING_LESS_085_FAULT
)
        {
          Serial.println( "REF(IN-) < 0.85 x V(BIAS), FORCE(-) is
open" );
        }
        else if( sensor5.status( ) & MAX31865_IS_RTDIN_LESS_085_FAULT
)
        {
          Serial.println( "RTD(IN-) < 0.85 x V(BIAS), FORCE(-) is
open" );
        }
        else if( sensor5.status( ) & MAX31865_IS_VOLTAGE_FAULT )
        {
          Serial.println( "Input voltage > V(DD) or < GND");
        }
        else
        {
          Serial.println( "Tuntematon virhe/unknown fault" );
        }
      }
       /*Sensor PT100 Nro 6-*/
      // jos fault status bitit ovat nollia, eli ei ole virheitä
```

```
      if( sensor6.status( ) == 0 )
      {
        // get temperature
        temperature = sensor6.lampotilanlasku( );
        // print results
        Serial.print( " T6 = "); Serial.print( temperature ); Se-
rial.println(" C" );
      }
      // on virheitä, faults
      else
      {
        Serial.print( "S6. Verheita, faults: " );
        Serial.print( sensor6.status( ) );
        Serial.print( ": " );
        // Datasheet table 7
        // find out faults
        if( sensor6.status( ) & MAX31865_IS_RTD_HIGH_THRESHOLD_FAULT )
        {
          Serial.println( "Measured resistance of PT100 greater than
HIGH Fault Threshold value" );
        }
        else if( sensor6.status( ) & MAX31865_IS_RTD_LOW_THRESH-
OLD_FAULT )
        {
          Serial.println( "Measured resistance of PT100 less than LOW
Fault Threshold value" );
        }
        else if( sensor6.status( ) & MAX31865_IS_REFIN_MORE_085_FAULT
)
        {
          Serial.println( "REF(IN-) > 0.85 x V(BIAS)" );
        }
        else if( sensor6.status( ) & MAX31865_IS_REFING_LESS_085_FAULT
)
        {
          Serial.println( "REF(IN-) < 0.85 x V(BIAS), FORCE(-) is
open" );
        }
        else if( sensor6.status( ) & MAX31865_IS_RTDIN_LESS_085_FAULT
)
        {
          Serial.println( "RTD(IN-) < 0.85 x V(BIAS), FORCE(-) is
open" );
        }
        else if( sensor6.status( ) & MAX31865_IS_VOLTAGE_FAULT )
        {
          Serial.println( "Input voltage > V(DD) or < GND");
        }
        else
        {
          Serial.println( "Tuntematon virhe/unknown fault" );
        }
      }


    delay( 3000 );
     if( Serial.read() == '0' )
        {
          // switch PWM off
          digitalWrite( FET_PIN, 0 );
          Serial.println("Mittaukset paattyivat");
```

```
        // digitalWrite( 13, LOW);
         break;
        }
    } // while( true ) lue....
  } // if letter == 1
} // loop
```

LIITE F: PC-YKSIKÖN OHJELMA

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;

using System.IO;
using System.IO.Ports;
using System.Text.RegularExpressions;

namespace PCM_readFromArduino
{
    public partial class Form2 : Form
    {
        public Form2()
        {
            InitializeComponent();
            openFileDialog1.InitialDirectory = AppDomain.Current-
Domain.BaseDirectory;
            getAvailablePorts();
        }

        // lisätään COM porttia
        void getAvailablePorts()
        {
            // saadaan tietoa
            String[] ports = SerialPort.GetPortNames();
            // tyhjennetään combobox
            PORT_nro.Items.Clear();
            PORT_nro.ResetText();
            // poistetaan samannimiset portit
            ports = ports.Distinct().ToArray();
            // laitetaan porttien nimet comboboxiin
            PORT_nro.Items.AddRange(ports);
            //PORT_nro.Items.AddRange(ports);
        }

        private void Form2_Load(object sender, EventArgs e)
        {
            // labels of chart
            chart1.ChartAreas[0].AxisX.Title = "Times(s)";
            chart1.ChartAreas[0].AxisY.Title = "Temperture C";
            chart1.ChartAreas[0].AxisX.TitleForeColor = System.Draw-
ing.Color.White;
            chart1.ChartAreas[0].AxisY.TitleForeColor = System.Draw-
ing.Color.White;

            // возможность масштаба
            chart1.ChartAreas[0].CursorX.IsUserEnabled = true;
            // включим масштаб по оси X
            chart1.ChartAreas[0].AxisX.ScaleView.Zoomable = true;
            // полоса прокрутки
            chart1.ChartAreas[0].AxisX.ScrollBar.IsPositionedInside =
true;
```

```
            // возможность выбора интервала для масштаба
            chart1.ChartAreas[0].CursorX.IsUserSelectionEnabled =
true;
            // возможность масштаба
            chart1.ChartAreas[0].CursorY.IsUserEnabled = true;
            // включим масштаб по оси Y
            chart1.ChartAreas[0].AxisY.ScaleView.Zoomable = true;
            // полоса прокрутки
            chart1.ChartAreas[0].AxisY.ScrollBar.IsPositionedInside =
true;
            // возможность выбора интервала для масштаба
            chart1.ChartAreas[0].CursorY.IsUserSelectionEnabled =
true;
            // getAvailablePorts();
            baud_rate.SelectedItem = "115200";
            label_status.Font = new Font( label_status.Font,
FontStyle.Italic );
            // для вывода приветствия
            DateTime nykyaika = DateTime.Now;
            if ( nykyaika.Hour >= 6 && nykyaika.Hour <= 11 )
                label_status.Text = "Huomenta! Good Morning!";
            if ( nykyaika.Hour >= 12 && nykyaika.Hour <= 15 )
                label_status.Text = "Päivää! Good afternoon!";
            if ( nykyaika.Hour >= 16 && nykyaika.Hour <= 23 )
                label_status.Text = "Hyvää iltaa! Good evening!";
            if ( nykyaika.Hour >= 23 && nykyaika.Hour <= 5 )
                label_status.Text = "Hyvää yötä! Good Night!";

            // button Save first disabled, nothing to save
            btn_save.Enabled = false;
            btn_csv.Enabled = false;
        }

        // axises of sensors
        int sensor_measure_nro = 0;
        int anturin_nro = 0;
        double anturin_arvo = 00.00;

        // temperature error of each sensor (6)
        double[] t_error = new double[6];

        // calibration time in seconds
        int calibr_seconds_left = 5;

        // values of sensors during calibration
        string calibr_val_sensor1 = "";
        string calibr_val_sensor2 = "";
        string calibr_val_sensor3 = "";
        string calibr_val_sensor4 = "";
        string calibr_val_sensor5 = "";
        string calibr_val_sensor6 = "";

        /*********************************
        * BUTTON MEASURE
        * 1. if timer is started
         * 1.1. Stop timer
         * 1.2. Enable buttons
         * 1.3. Close serial connection
        * 2. if timer is not started
         * 2.1. is port name chosen
         * 2.2. if yes
```

```
         * 2.3. set port name, speed
         * 2.4. Disable buttons
         * 2.5. Start timer
        */
        private void btn_measure_Click(object sender, EventArgs e)
        {
            // if timer is enabled
            if ( timer1.Enabled )
            {
                // disable timer
                timer1.Enabled = false;

                // change buttons text, styles etc
                btn_measure.Text = "MITTAA / MEASURE";
                btn_measure.Enabled = false;
                label_status.Text = "Mittaus päättyi / Masurement fin-
ished";
                btn_save.Enabled = true;
                btn_load.Enabled = true;
                btn_csv.Enabled = true;
                btn_restart.Enabled = true;
                btn_exit.Enabled = true;

                // close serial connection
                serialPort1.WriteLine("0");
                int i = 0;
                while (i < 100)
                {
                    serialPort1.WriteLine("0");
                    i++;
                }
                // close port
                serialPort1.Close();

            }
            // if timer is disabled
            else
            {
                // check port name, проверяем, выбран ли порт
                if ( PORT_nro.Text == "" )
                {
                    label_status.Text = "Valitse PORT / Choose PORT";
                    label_status.ForeColor = System.Drawing.Color.To-
mato;
                    groupBox2.ForeColor = System.Drawing.Color.Tomato;
                }
                else
                {
                    // port name
                    serialPort1.PortName = PORT_nro.Text;
                    // connection speed
                    serialPort1.BaudRate = Convert.ToInt32(
baud_rate.Text );

                    // launch COM port and start communication with
Arduino, command 1, запускаем COM
                    serialPort1.Open();
                    serialPort1.WriteLine("1");
```

```
                    // clear charts, if char is already opened, clear
for new drawing
                    for ( int i = 0; i < 6; i++ )
                        chart1.Series[i].Points.Clear();

                    // status, статус сообщение
                    label_status.ForeColor = System.Draw-
ing.Color.SpringGreen;
                    label_status.Text = "Mittaus on käynnissä / Meas-
urement in progress";
                    groupBox2.ForeColor = System.Drawing.Color.Silver;

                    // enable timer, Таймер запускаем
                    timer1.Enabled = true;

                    // change buttons text, styles etc
                    btn_measure.Text = "LOPETA / STOP";
                    btn_use_calibration.Enabled = false;
                    btn_save.Enabled = false;
                    btn_load.Enabled = false;
                    btn_csv.Enabled = false;
                    btn_calibrate.Enabled = false;
                    btn_restart.Enabled = false;
                    btn_exit.Enabled = false;

                } // порт выбран тогда
            }   // if timer is disabled, если выключен Таймер
        } // BUTTON MEASURE
        /*****************************
        * TIMER MEASURE
        * 1. get line from Arduino
        * 2. Check line and get data from it
        * 3. Draw chart
        */

        private void timer1_Tick(object sender, EventArgs e)
        {
            // get the line, считаем строку
            string rivi = "";
            rivi = serialPort1.ReadLine();

            // delete all spaces
            rivi = rivi.Trim();

            // is the line empty, не пустая ли строка
            if ( rivi.Trim(' ') != "" )
            {
                // get the first letter
                string eka_kirjain = rivi.Substring(0, 1);
                //    if it is like #1 or #2
                if ( eka_kirjain == "#" )
                {
                    // delete # symbol for number of measure
                    sensor_measure_nro = Convert.ToInt16( rivi.Re-
place( "#", string.Empty ) );
                }
                //    if it is like T1 or T2
                if ( eka_kirjain == "T" )
                {
                    // delete all spaces within line, now T1=0 or
T1=22.5 or T1=22,5
```

```
                    rivi = rivi.Replace( " ", string.Empty );

                    // get the number of sensor
                    anturin_nro = Convert.ToInt16( rivi.Substring( 1,
1 ) );
                    // delete T1=, now 0 or 22.5 or 22,5
                    rivi = rivi.Replace( "T" + anturin_nro + "=",
string.Empty );
                    // is it like 24.5
                    if ( rivi.Contains('.') )
                    {
                        // change it to 24,5
                        rivi = rivi.Replace( '.', ',' );

                        // is it more than 10,0
                        if ( Convert.ToDouble( rivi ) >= 10 )
                        {
                            // 1 + 0 + , + 0
                            anturin_arvo = Convert.ToDouble( (
rivi.Substring( 0, 4 ) ) );
                        }
                        // is it like 9,0
                        else
                        {
                            // 9 + , + 9
                            anturin_arvo = Convert.ToDouble( (
rivi.Substring( 0, 3 ) ) );
                        }
                    }
                    else // if Contains ,
                    {
                        // is it like 24,5
                        if ( rivi.Contains(',') )
                        {
                            // is it more than 10,0
                            if ( Convert.ToDouble( rivi ) >= 10 )
                            {
                                // 1 + 0 + , + 0
                                anturin_arvo = Convert.ToDouble(
rivi.Substring( 0, 4 ) );
                            }
                            // is it like 9,0
                            else
                            {
                                // 9 + , + 9
                                anturin_arvo = Convert.ToDouble(
rivi.Substring( 0, 3 ) );
                            }
                        }
                        // it is like 24 or 0
                        else
                        {
                            anturin_arvo = Convert.ToDouble( rivi.Re-
place( "T" + anturin_nro + "=", string.Empty ) );

                            // is it more than 10
                            if ( Convert.ToDouble( rivi ) >= 10 )
                            {
                                // 1 + 0
                                anturin_arvo = Convert.ToDouble( (
rivi.Substring( 0, 2 ) ).Replace( '.', ',') );
```

```
                            }
                            // is it like 9
                            else
                            {
                                // 9
                                anturin_arvo = Convert.ToDouble( (
rivi.Substring( 0, 1 ) ).Replace( '.', ',') );
                            }
                        } // it is like 24 or 0

                    } // if Contains ,
                    // draw a point
                    // use calibration values!! t_error
                    chart1.Series[anturin_nro - 1].Points.AddXY(sen-
sor_measure_nro, anturin_arvo - t_error[anturin_nro - 1]);
                } // eka_kirjain == "T"
            } // is the line empty
        } // TIMER MEASURE

        /*****************************
         * BUTTON LOAD
         * 1. Ger information of each sensor
         * 2. Read it correctly
         * 3. Draw charts
         */
        private void btn_load_Click(object sender, EventArgs e)
        {

            // is FileDialog open
            if (openFileDialog1.ShowDialog() == DialogResult.OK)
            {
                // variable for reading files
                StreamReader arvotdiedostosta = new StreamReader(open-
FileDialog1.FileName);

                // X axises of each sensor
                int sensor_measure_nro = 0;
                // sensor number
                int anturin_nro = 0;
                // current sensor's value
                double anturin_arvo = 00.00;
               // if char is already opened, clear for new drawing
                for (int i = 0; i < 6; i++)
                    chart1.Series[i].Points.Clear();

                // read until end of file
                while (!arvotdiedostosta.EndOfStream)
                {
                    // get the line, считаем строку
                    string rivi = arvotdiedostosta.ReadLine();

                    // delete all spaces
                    rivi = rivi.Trim();
                    // is the line empty, не пустая ли строка
                    if (rivi.Trim(' ') != "")
                    {
                        // get the first letter
                        string eka_kirjain = rivi.Substring(0, 1);
                        //    if it is like #1 or #2
                        if (eka_kirjain == "#")
                        {
```

```
                            // delete # symbol for number of measure
                            sensor_measure_nro = Con-
vert.ToInt16(rivi.Replace("#", string.Empty));
                        }
                        //    if it is like T1 or T2
                        if (eka_kirjain == "T")
                        {
                            // delete all spaces within line, now T1=0
or T1=22.5 or T1=22,5
                            rivi = rivi.Replace(" ", string.Empty);


                            // get the number of sensor
                            anturin_nro = Convert.ToInt16(rivi.Sub-
string(1, 1));

                            // delete T1=, now 0 or 22.5 or 22,5
                            rivi = rivi.Replace("T" + anturin_nro +
"=", string.Empty);

                            // is it like 24.5
                            if (rivi.Contains('.'))
                            {
                                // change it to 24,5
                                rivi = rivi.Replace('.', ',');

                                // is it more than 10,0
                                if (Convert.ToDouble(rivi) >= 10)
                                {
                                    // 1 + 0 + , + 0
                anturin_arvo = Convert.ToDouble((rivi.Substring(0, 4)));
                                }
                                // is it like 9,0
                                else
                                {
                                    // 9 + , + 9
                anturin_arvo = Convert.ToDouble((rivi.Substring(0, 3)));
                                }

                            }      // if Contains .
                            else
                            {
                                // is it like 24,5
                                if (rivi.Contains(','))
                                {
                                    // is it more than 10,0
                                    if (Convert.ToDouble(rivi) >= 10)
                                    {
                                        // 1 + 0 + , + 0
                 anturin_arvo = Convert.ToDouble(rivi.Substring(0, 4));
                                    }
                                    // is it like 9,0
                                    else
                                    {
                                        // 9 + , + 9
                 anturin_arvo = Convert.ToDouble(rivi.Substring(0, 3));
                                    }
                                }
                                // it is like 24 or 0
                                else
                                {
```

```
                                    anturin_arvo = Convert.ToDou-
ble(rivi.Replace("T" + anturin_nro + "=", string.Empty));

                                    // is it more than 10
                                    if (Convert.ToDouble(rivi) >= 10)
                                    {
                                        // 1 + 0
                                        anturin_arvo = Convert.ToDou-
ble((rivi.Substring(0, 2)).Replace('.', ','));
                                    }
                                    // is it like 9
                                    else
                                    {
                                        // 9
                                        anturin_arvo = Convert.ToDou-
ble((rivi.Substring(0, 1)).Replace('.', ','));
                                    }
                                } // // it is like 24 or 0

                            } // else
                            // draw a point
                            chart1.Series[anturin_nro -
1].Points.AddXY(sensor_measure_nro, anturin_arvo);
                        }
                    } // is the line empty
                } // while read file end
                // поле загрузки файл, очищаем память от потока
                arvotdiedostosta.Close();
            } // if openFileDialog1

            // button save is enabled now
            if (btn_save.Enabled == false)
                btn_save.Enabled = true;
            if (btn_csv.Enabled == false)
                btn_csv.Enabled = true;

            // disable otherwise
            btn_measure.Enabled = false;
            btn_calibrate.Enabled = false;
            btn_use_calibration.Enabled = false;
        } // BUTTON LOAD

        /*****************************
         * BUTTON SAVE
         * 1. Get time for file name
         * 2. Count points of charts
         * 3. Save sensors' data to txt
         */
        private void btn_save_Click(object sender, EventArgs e)
        {
            // get current time
            DateTime nykyaika = DateTime.Now;

            // open fileDialog and offer a simple name of a file
            saveFileDialog1.FileName = "Measurement_" +
nykyaika.ToString("yyyy-MM-dd HH-mm-ss") + ".txt";

            // if fileDialog is open
            if ( saveFileDialog1.ShowDialog() == DialogResult.OK )
            {
                // variable for writting data
```

```
                StreamWriter data_to_file = new StreamWriter( save-
FileDialog1.FileName );

                // get number of measurements
                int min_points_of_all_charts =
Math.Min(chart1.Series[0].Points.Count,

Math.Min(chart1.Series[1].Points.Count,

Math.Min(chart1.Series[2].Points.Count,

Math.Min(chart1.Series[3].Points.Count,

Math.Min(chart1.Series[4].Points.Count,
                                                    chart1.Se-
ries[5].Points.Count))))));

                // save all data, axis X
                for (int j = 0; j < min_points_of_all_charts; j++)
                {
                    // print number of measurement
                    data_to_file.WriteLine( "#" + ( j + 1 ).ToString()
);
                    // axis Y, 6 sensors
                    for (int i = 0; i < 6; i++)
                    {
                        // print data of points
                        data_to_file.Write("T" + (i + 1).ToString() +
" = ");
                        data_to_file.WriteLine(chart1.Se-
ries[i].Points[j].YValues[0]);
                    }
                }
                // complete sending
                data_to_file.Close();
            } // fileDialog
        } // BUTTON SAVE

        /*****************************
       * BUTTON SAVE CSV
       * 1. Save to csv file
       */
        private async void btn_csv_Click(object sender, EventArgs e)
        {
            // get current time
            DateTime nykyaika = DateTime.Now;

            // open fileDialog and offer a simple name of a file
            saveFileDialog2.FileName = "MeasurementCSV_" +
nykyaika.ToString("yyyy-MM-dd HH-mm-ss") + ".csv";

            // if fileDialog is open
            if ( saveFileDialog2.ShowDialog() == DialogResult.OK )
            {
                using(StreamWriter sw = new StreamWriter(new
FileStream(saveFileDialog2.FileName, FileMode.Create), Encoding.UTF8))
                {
                    StringBuilder sb = new StringBuilder();
                    sb.AppendLine("Sensor1; Sensor2; Sensor3; Sensor4;
Sensor5; Sensor6");
```

```
                    // get number of measurements
                    int min_points_of_all_charts =
Math.Min(chart1.Series[0].Points.Count,

Math.Min(chart1.Series[1].Points.Count,

Math.Min(chart1.Series[2].Points.Count,

Math.Min(chart1.Series[3].Points.Count,

Math.Min(chart1.Series[4].Points.Count,
                                                    chart1.Se-
ries[5].Points.Count)))));

                    double[] row_values = new double[6];

                    // save all data, axis X
                    for (int j = 0; j < min_points_of_all_charts; j++)
                    {
                        // axis Y, 6 sensors
                        for (int i = 0; i < 6; i++)
                        {
                            // print data of points
                            row_values[i] = chart1.Se-
ries[i].Points[j].YValues[0];
                        }
                        sb.AppendLine(string.Format("{0}; {1}; {2};
{3}; {4}; {5}", row_values[0].ToString(), row_values[1].ToString(),
row_values[2].ToString(), row_values[3].ToString(), row_val-
ues[4].ToString(), row_values[5].ToString()));
                    }
                    await sw.WriteLineAsync(sb.ToString());
                }
            } // fileDialog
        } // BUTTON SAVE CSV
        /*****************************
         * BUTTON CALIBRATE
         * 1. Check COM PORT communication information
         * 2. Disable buttons
         * 3. Start serial Port
         * 4. Start timer for calibration measurements
         */
        private void btn_calibrate_Click(object sender, EventArgs e)
        {
            // is Port chosen, проверяем, выбран ли порт
            if ( PORT_nro.Text == "" )
            {
                // status information
                label_status.Text = "Valitse / Select PORT";
                label_status.ForeColor = System.Drawing.Color.Tomato;
                groupBox2.ForeColor = System.Drawing.Color.Tomato;
            }
            // COM port is chose, порт выбран тогда
            else
            {
                // start calibration or not?
                if( MessageBox.Show("Aloitetaanko kalibrointi? Start
calibration?",
                                    "Kalibrointi / Calibration",
                                    MessageBoxButtons.YesNo,
                                    MessageBoxIcon.Question )
```

```
                                        == DialogResult.Yes )
                {
                    // status information, статус сообщение
                    label_status.ForeColor = System.Draw-
ing.Color.SpringGreen;
                    label_status.Text = "Kalibrointi on käynnissä /
Calibration in progress";
                    groupBox2.ForeColor = System.Drawing.Color.Silver;

                    // color of label is green
                    label_calibration_time.ForeColor = System.Draw-
ing.Color.SpringGreen;

                    // disable useless buttons
                    btn_use_calibration.Enabled = false;
                    btn_measure.Enabled = false;
                    btn_save.Enabled = false;
                    btn_load.Enabled = false;
                    btn_csv.Enabled = false;
                    btn_restart.Enabled = false;
                    btn_calibrate.Enabled = false;

                    // feed serial name and speed,
                    serialPort1.PortName = PORT_nro.Text;
                    serialPort1.BaudRate = Con-
vert.ToInt32(baud_rate.Text);

                    // launch serial connection, запускаем COM
                    serialPort1.Close();
                    serialPort1.Open();

                    // timer start for calbration
                    timer_calibration.Start();
                }  // if (  start calibration or not?  )
            } // else
        } // BUTTON CALIBRATE

        /************************************
         * TIMER_FOR_CALIBRATION_MEASUREMENTS
         * 1. Start calibration time
         * 2. If time is up
         * 2.1. Start communication with Arduino
         * 2.2. Get sensor values
         * 2.3. Show results
         * 2.4. If accept results
         * 2.4.1 Save results to file
         */
        private void timer_calibration_Tick(object sender, EventArgs
e)
        {

            // label calibration time left
            label_calibration_time.Text = calibr_sec-
onds_left.ToString() + " sec";

            // time is up
            if ( calibr_seconds_left == 0 )
            {
                // disable timer
                timer_calibration.Stop();
```

```
                // show message box
                MessageBox.Show("Valmis, Ready", "Aika, Time");

                label_status.Text = "Kalibrointi päättyi / Calibration
is finished";

                // send command to Arduino to start measurements
                serialPort1.WriteLine("1");

                // read data
                while ( true )
                {
                    // get the line
                    string rivi ="";
                    rivi = serialPort1.ReadLine();

                    // delete all spaces,
                    rivi = rivi.Trim();

                    // is the line empty, не пустая ли строка
                    if ( rivi.Trim(' ') != "" )
                    {
                        // get the first symbol
                        string eka_kirjain = rivi.Substring(0, 1);
                        //    if it is like #1 or #2
                        if ( eka_kirjain == "#" )
                        {
                            // delete # symbol for number of measure
                            sensor_measure_nro = Convert.ToInt16(
rivi.Replace( "#", string.Empty ) );
                        }
                        //    if it is like T1 or T2
                        if ( eka_kirjain == "T" )
                        {
                            // delete all spaces within line, now T1=0
or T1=22.5 or T1=22,5
                            rivi = rivi.Replace( " ", string.Empty );

                            // get the number of sensor
                            anturin_nro = Convert.ToInt16( rivi.Sub-
string( 1, 1 ) );

                            // delete T1=, now 0 or 22.5 or 22,5
                            rivi = rivi.Replace( "T" + anturin_nro +
"=", string.Empty );

                            // is it like 24.5
                            if ( rivi.Contains('.') )
                            {

                                // change it to 24,5
                                rivi = rivi.Replace( '.', ',' );

                                // is it more than 10,0
                                if ( Convert.ToDouble( rivi ) >= 10 )
                                {
                                    // 1 + 0 + , + 0
                                    anturin_arvo = Convert.ToDouble( (
rivi.Substring(0, 4) ) );
                                }
                                // is it like 9,0
```

```
                                else
                                {
                                    // 9 + , + 9
                                    anturin_arvo = Convert.ToDouble( (
rivi.Substring(0, 3) ) );
                                }

                            } // if Contains .
                            else
                            {
                                // is it like 24,5
                                if ( rivi.Contains(',') )
                                {
                                    // is it more than 10,0
                                    if ( Convert.ToDouble( rivi ) >=
10 )
                                    {
                                        // 1 + 0 + , + 0
                                        anturin_arvo = Convert.ToDou-
ble( rivi.Substring( 0, 4 ) );
                                    }
                                    // is it like 9,0
                                    else
                                    {
                                        // 9 + , + 9
                                        anturin_arvo = Convert.ToDou-
ble( rivi.Substring( 0, 3 ) );
                                    }
                                } // is it like 24,5

                                else // it is like 24 or 0
                                {

                                    anturin_arvo = Convert.ToDouble(
rivi.Replace( "T" + anturin_nro + "=", string.Empty ) );

                                    // is it more than 10
                                    if (Convert.ToDouble(rivi) >= 10)
                                    {
                                        // 1 + 0
                                        anturin_arvo = Convert.ToDou-
ble( ( rivi.Substring(0, 2) ).Replace('.', ','));
                                    }
                                    // is it like 9
                                    else
                                    {
                                        // 9
                                        anturin_arvo = Convert.ToDou-
ble( ( rivi.Substring(0, 1) ).Replace('.', ',') );
                                    }
                                } // // it is like 24 or 0

                            } // else

                            // feed values of sensors in calibration
                            switch( anturin_nro )
                            {
                                case 1:
                        calibr_val_sensor1 = anturin_arvo.ToString();
                                    break;
                                case 2:
```

```
                        calibr_val_sensor2 = anturin_arvo.ToString();
                                    break;
                                case 3:
                        calibr_val_sensor3 = anturin_arvo.ToString();
                                    break;
                                case 4:
                        calibr_val_sensor4 = anturin_arvo.ToString();
                                    break;
                                case 5:
                        calibr_val_sensor5 = anturin_arvo.ToString();
                                    break;
                                case 6:
                        calibr_val_sensor6 = anturin_arvo.ToString();
                                    break;
                            }

                            // check, values not empty
                            if ( calibr_val_sensor1 != "" && cal-
ibr_val_sensor6 != "" )
                            {
                                // print values
                                MessageBox.Show("T1 = " + cal-
ibr_val_sensor1 +
                                        " | T2 = " + calibr_val_sensor2 +
                                        " | T3 = " + calibr_val_sensor3 +
                                        " | T4 = " + calibr_val_sensor4 +
                                        " | T5 = " + calibr_val_sensor5 +
                                    " | T6 = " + calibr_val_sensor6 + " ");
                                // end of measurements
                                break;
                            } // check, values not empty
                        } // if eka kirjain T
                    } // if rivi trim
                } // while true

                // send command to Arduino to finish measurements
                 serialPort1.WriteLine("0");

                // show dialog, Save results?
                if ( MessageBox.Show("Tallennetaanko tulokset? Save
results?",
                                        "Kalibrointi / Calibration",
                                        MessageBoxButtons.YesNo,
                                        MessageBoxIcon.Question) == Di-
alogResult.Yes )
                {
                    // get current time
                    DateTime nykyaika = DateTime.Now;
                    // open fileDialog and offer a simple name of a
file
                    saveFileDialog1.FileName = "Calibration_" +
nykyaika.ToString("yyyy-MM-dd HH-mm-ss") + ".txt";

                    // if fileDialog is open
                    if ( saveFileDialog1.ShowDialog() == Di-
alogResult.OK )
                    {
                        // variable for writting data
                        StreamWriter data_to_file = new Stream-
Writer(saveFileDialog1.FileName);
```

```
                        // write data to file
                        data_to_file.WriteLine( calibr_val_sensor1 );
                        data_to_file.WriteLine( calibr_val_sensor2 );
                        data_to_file.WriteLine( calibr_val_sensor3 );
                        data_to_file.WriteLine( calibr_val_sensor4 );
                        data_to_file.WriteLine( calibr_val_sensor5 );
                        data_to_file.WriteLine( calibr_val_sensor6 );
                        // complete sending
                        data_to_file.Close();
                    } // fileDialog
                    // enable Restart button
                    btn_restart.Enabled = true;
                } // messgebox
                else
                {
                    // enable Restart button
                    btn_restart.Enabled = true;
                }

            } // // time is up
            else // time is not up
            {
                // reduce 1 second
                calibr_seconds_left--;
            }
        } // TIMER FOR CALIBRATION MEASUREMENTS
         /**********************************
         * BUTTON USE CALIBRATION
         * 1. Open file with calibration information
         * 2. Read file
         * 3. Find errors
         * 4. Feed to array "t_error"
         */
        private void btn_use_calibration_Click(object sender, Even-
tArgs e)
        {
            // is FileDialog open,
            if ( openFileDialog1.ShowDialog() == DialogResult.OK )
            {
                // variable for reading files,
                StreamReader arvotdiedostosta = new StreamReader(open-
FileDialog1.FileName);

                // variable for sensor number (6)
                int sensor_nro = 0;

                // read until end of file,
                while ( !arvotdiedostosta.EndOfStream )
                {
                    // get the line, считаем строку
                    string rivi = arvotdiedostosta.ReadLine();

                    // delete all spaces,
                    rivi = rivi.Trim();
                    // is the line empty, не пустая ли строка и 1ы
символ число
                    if ( rivi.Trim(' ') != "" && (Convert.To-
Char(rivi.Substring(0, 1))<='9') && (Convert.ToChar(rivi.Substring(0,
1))>='1') )
                    {
```

```
                        // Feed temperature values to array "t_error",
заносим только значение датчиков
                        t_error[sensor_nro] = Convert.ToDouble(rivi);

                    } // is the line empty
                    else
                    {
                        break;
                    }
                    // next sensor
                    sensor_nro++;
                } // while read file end
                // close file,
                arvotdiedostosta.Close();

            } // if openFileDialog1
            // find the most common value
            double most_common_value = t_error.GroupBy(v => v)
            .OrderByDescending(g => g.Count())
            .First()
            .Key;
             MessageBox.Show("most_common_value = " + most_com-
mon_value.ToString());

            // feed errors of sensor values, заносим уже погрешность
            for (int i = 0; i < 6; i++ )
            {
                // count errors and round values, считам и округляем
                t_error[i] = Math.Round( most_common_value - t_er-
ror[i],2);
                 MessageBox.Show("t_error[" + i.ToString() + "] = " +
t_error[i].ToString());
            }

        } // BUTTON USE CALIBRATION
        /*******************************
         * BUTTON Reset
         * Reload all applicatio
         */
        private void btn_restart_Click_1(object sender, EventArgs e)
        {
            // 1. start new instance of application
            System.Diagnostics.Process.Start(Application.Executa-
blePath);
            // 2. turn off current app
            this.Close();
        } // BUTTON Reset
        /*******************************
         * BUTTON EXIT
         * 1. Check mesurements in progress
         * 2. Close serial port
         * 3. Close application
         */
        private void btn_exit_Click(object sender, EventArgs e)
        {
            // if measurements in progress
            if (btn_measure.Text == "KESKEYTA")
                // send command to Arduino to stop measurements
                serialPort1.WriteLine("0");

            // close serial port
```

```
            serialPort1.Close();

            // close application
            Application.Exit();
        }// BUTTON EXIT
    }
}
```